PHASE-TYPE APPROXIMATIONS FOR WEAR PROCESSES IN A SEMI-MARKOV ENVIRONMENT

THESIS

Christopher J. Solo, Captain, USAF

AFIT/GOR/ENS/04-11

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense or the United States Government.

AFIT/GOR/ENS/04-11

PHASE-TYPE APPROXIMATIONS FOR WEAR PROCESSES IN A SEMI-MARKOV ENVIRONMENT

THESIS

Presented to the Faculty
Department of Operational Sciences
Graduate School of Engineering and Management
Air Force Institute of Technology
Air University
Air Education and Training Command
in Partial Fulfillment of the Requirements for the
Degree of Master of Science in Operations Research

Christopher J. Solo, B.S.
Captain, USAF

March 2004

AFIT/GOR/ENS/04-11

PHASE-TYPE APPROXIMATIONS FOR WEAR PROCESSES IN A SEMI-MARKOV ENVIRONMENT

Christopher J. Solo, B.S.

Captain, USAF

Approved:

//Signed//	12 March 2004
Dr. Jeffrey P. Kharoufeh Thesis Advisor	Date
//Signed//	12 March 2004
Dr. Sharif Melouk Committee Member	Date

AFIT/GOR/ENS/04-11

Abstract

The reliability of a single-unit system experiencing degradation (wear) due to the influence of a general, observable environment process is considered. In particular, the failure time distribution is evaluated using only observations of the unit's current operating environment which is characterized as a finite semi-Markov process (SMP). In order to impose the Markov property, generally distributed environment state sojourn times are approximated as phase-type (PH) random variables using observations of state holding times and transition rates. The use of PH distributions facilitates the use of existing analytical results for reliability evaluation of units subject to an environment process that evolves as a continuous-time Markov chain. The procedure is illustrated through three numerical examples, and results are compared with those obtained via Monte Carlo simulation. The maximum absolute deviation in probability for failure time distributions was on the order of 0.004. The results of this thesis provide a novel approach to the reliability analysis of units operating in randomly evolving environments for which degradation or failure time observations are difficult or impossible to obtain.

Acknowledgements

I would like to acknowledge those who were instrumental in my completion of this research effort. First, I extend many thanks to my thesis advisor, Dr. Jeffrey Kharoufeh, for providing the guidance and motivation crucial to my understanding, development of, and solution to this thesis problem. Without Dr. Kharoufeh's vast expertise and keen insight, this thesis would most certainly not have been possible. Next, I thank Major Steve Cox for navigating me through computational details and providing numerous tutorials on the subject matter. And most importantly, I thank my wife for her remarkable support during the countless hours I spent away from her and our two young children during the length of this research effort. I am truly grateful to all three of them and owe them many hours of attention and playtime.

Christopher J. Solo

Table of Contents

List of Figures

List of Tables

PHASE-TYPE APPROXIMATIONS FOR WEAR PROCESSES IN A SEMI-MARKOV ENVIRONMENT

1. Introduction

1.1 Background

Classical reliability analysis involves the estimation of the probability distribution of component lifetimes based on historical observations of failure time. This has been the most thoroughly studied form of reliability analysis, and several standard models have been used for decades in the design, manufacture, and maintenance of components and systems. As an example, the exponential probability distribution has been widely accepted as an estimate for the lifetime distribution of electronic components. However, advances in technology and production methods have made it impractical to observe failures of systems with very long lifetimes. Additionally, the observation of failure times may be a prohibitively expensive approach given the complexity and high costs of modern systems. For instance, consider a satellite operating in orbit. Evaluating the reliability of a single component or the entire system by running until failure is clearly an infeasible approach. An alternative technique commonly applied to alleviate these shortfalls in failure time estimation is accelerated lifetime testing. This technique involves lifetime testing of components under accelerated laboratory conditions. However, the value of this approach may be countered by the fact that such artificial laboratory tests often do not accurately represent the realistic conditions of a unit's operating environment.

Extremely long lifetimes, high costs, and unrealistic testing environments are not the only drawbacks of failure-based reliability analysis. A system's reliability may not necessarily be defined as the time of a catastrophic failure–achievement of a

threshold level of wear or degradation may signal the need for maintenance or the end of a system's useful operating life. With failure-based reliability analysis, these measures are not necessarily observed. In an effort to overcome these shortfalls, a more recent trend in system lifetime estimation has been the application of degradation-based reliability techniques. In contrast to failure-based approaches, these techniques involve lifetime estimation of a component or system based on observations of cumulative degradation over time. However, degradation measurements may be difficult or even impossible to obtain. For instance, it may be infeasible or prohibitively expensive to record degradation measurements for a solar panel attached to an on-orbit satellite. Although this technique may more accurately predict the degradation path leading to system failure, it does not necessarily take into account the ambient environment that is causing the unit to degrade.

In such cases as the on-orbit satellite, the environment to which the system is exposed is of great importance. For instance, differing levels of ultraviolet radiation may have varied effects on the overall wear of the solar panel. An environment-based approach to reliability analysis seeks to account for the impact the surrounding environment has on the operation and degradation of the system under study. Since very few systems operate in complete isolation, environment-based approaches are valuable in evaluating reliability under realistic operating conditions. In lieu of imposing an artificial setting, environment-based reliability models attempt to study both the operating environment and its effects on the system. As with accelerated testing techniques, this approach's success in estimating system reliability depends on the level of accuracy of the environment model under which the system is studied and the effect of various environment conditions on the degradation of the system. If the environment consists of easily recognizable and definable discrete states, modelling of the environment may be a fairly straightforward process. However, characterizing and defining the various states of a continuous environment process may not be an easy task. Furthermore, in some cases, the transitions from one environment state

to another are not easily observable. Therefore, another potential drawback of an environment-based reliability approach is in the discretization of its feasible state space. Finally, it may be difficult or even impossible to observe the environment. Systems operating under deep water or in outer space are examples in which the environment may be difficult to characterize.

Despite the drawbacks noted above, environment-based reliability models have potential for accurate lifetime estimation of systems operating in known or observable environments. Several techniques, both probabilistic and statistical in nature, have been developed to evaluate the reliability, and particularly the failure time distribution, of a system experiencing wear due to a dynamic environment process. According to Singpurwalla [25], an advantage of probabilistic approaches, via the use of stochastic models, is that both the physics of system or component failure and the dynamic nature of the environment are taken into account. Following this approach, Kharoufeh [16] studied the reliability (via derivation of the failure time distribution) of a single-unit system experiencing wear due to a multi-state stochastic environment process. This technique uses a Markov additive process (MAP), as reviewed by Singpuwalla [25]. Using the MAP as the basis for the overall model, the cumulative damage of the system is modelled as a continuous, nondecreasing stochastic process, and the random environment process is modelled as a finite state space stochastic process. In [16], Kharoufeh assumed the environment process to be a continuous-time Markov chain (CTMC) in which the evolution of the environment depends only upon the current state, and the time spent in each state is assumed to be an exponentially distributed random variable. However, this model's assumption of exponentially distributed state holding times limits its real-world applicability. In many situations, the distribution of the random time spent in each state of the environment may be either nonexponential or simply unknown. A probabilistic technique based solely on measurements of the environment state holding times would alleviate the restrictive need for exponential (or any other) state holding time dis-

tributions. This thesis effort, based on the environment-based approach, attempts to address this shortcoming by presenting numerical approximation procedures for general environment models.

1.2 Problem Definition and Methodology

This thesis proposes a method for estimating the lifetime of a system based solely on observations of the environment to which the system is exposed and knowledge of the environment's impact on the degradation of the system. In particular, the reliability of a single-unit system experiencing nondecreasing wear due to a multi-state environment process in which the evolution of the environment process depends only upon its current state is investigated. Furthermore, it is assumed that the time spent in each state is known only to be a positive-valued, nondefective random variable. That is, there is zero probability that the environment process will remain in any one state indefinitely.

This work will build upon and generalize the results of [16] and [17]. In those works, the authors assumed the wear-inducing environment to be a continuous-time Markov chain (CTMC). As such, the next state of the environment depends exclusively on the current state, forming an embedded discrete-time Markov chain (DTMC). Additionally, the time spent in each of the various environment states is assumed to be an exponentially distributed random variable. In reality, state holding time distributions may not follow exponential distributions at all. In fact, the particular state holding time distributions may be completely unknown. However, it may be assumed that transitions to future states of the environment are still dependent only upon the current state, thus retaining an embedded DTMC. In such cases, the environment process may be characterized as a semi-Markov process. In this work, the primary focus is the reliability analysis of systems experiencing cumulative wear damage due to the influence of a dynamic external environment modelled

as a finite state space semi-Markov process which allows for general state holding time distributions.

The case of nonexponential environment state holding times is studied by approximating arbitrary distributions with phase-type (PH) distributions. That is, the arbitrary holding time distributions are approximated as the time to absorption of an underlying Markov process that must be estimated. In order to construct such approximations, the environment process must be observed for some period of time, and the time spent in each environment state must be recorded. Furthermore, the number of transitions among the various states must be observed and recorded in order to statistically estimate the inter-state transition rates. The advantage of PH distributions is that they retain the Markov property, supplanting the need for exponential state holding time distributions. This thesis will demonstrate that, by imposing the Markovian property by means of phase-type approximations, the semi-Markov environment process can be converted to a CTMC, and the results developed by Kharoufeh [16] and Kharoufeh and Sipe [17] can be applied.

The results of this research can be applied to the reliability analysis of components or systems that degrade over time due to exposure to varying levels of temperature, pressure, or some other environmental measure. This thesis effort will be valuable to those who design, evaluate, repair, or replace systems exposed to random environments. In particular, designers of components used aboard satellites or underwater vessels, systems known to operate in extreme environments, may find the technique developed in this thesis to be particularly useful. This work will also advance the current state-of-the-art in estimation techniques via stochastic modelling of degradation and environment processes.

1.3 Outline of the Thesis

The remainder of this thesis is organized as follows. Chapter 2 reviews the current literature on several probabilistic and statistical degradation-based reliability analysis techniques. Chapter 3 describes the formal mathematical model, presents the techniques used to construct phase-type approximations of arbitrary distributions, and proposes a method by which single-unit lifetime distributions may be approximated in semi-Markovian environments. Chapter 4 compares the failure time distribution results of numerical examples of the proposed technique with those obtained via simulation. Finally, chapter 5 presents conclusions and some suggestions for future research.

2. Review of the Literature

This chapter provides an overview of the literature concerning techniques for the analysis of degradation- or environment-based reliability. Although certainly not an exhaustive survey, it attempts to provide a sense of the two prominent methods for evaluating the lifetime of systems experiencing wear over time: probabilistic and statistical approaches.

2.1 Probabilistic Approaches to Degradation-Based Reliability

Several probabilistic approaches to degradation-based reliability are found in the literature. In his extensive survey of probabilistic approaches to failure time analysis, Singpurwalla [25] considers both the physics of failure and the characteristics of the operating environment. In general, the variation of the degradation rate is caused by a dynamic environment; this environment can be described by a stochastic process. However, failure time distributions derived from stochastic processes often do not have closed-form expressions and can only be expressed via Laplace transforms, as seen in [16]. Singpurwalla [25] notes four broad strategies that have been developed for modelling failure distributions based on stochastic processes:

1. Wear is described by a stochastic process with continuous sample path (i.e. a diffusion process), such as Wiener, gamma, or deterministic;
2. failure rate is described by a stochastic process, such as gamma, shot-noise, functionals of a Wiener process, or a Lévy process;
3. the environment is described by a stochastic process, typically a shock-inflicting Poisson process;
4. a response variable correlated with the lifelength is described by a stochastic process, such as a stationary, continuous-time Gaussian process.

When wear is modelled as a stochastic process with continuous sample path, Singpurwalla [25] notes that use of a Wiener process to describe diffusion of the item state may provide better goodness-of-fit than use of the exponential and Weibull distributions. In [25], the author describes such a process, also known as Brownian motion, as follows. The process $\{\gamma(s), s \geq 0\}$ is such that $\gamma(0) = 0, \gamma(s)$ has stationary independent incremements, and, for every $s > 0$, $\gamma(s) \sim \mathcal{N}(0, c^2 s)$ for some $c > 0$. Singpurwalla [25] adds that the Wiener process is a Markov process since the independent increments property of $\{\gamma(s), s \geq 0\}$ implies that, for any $t > 0$,

$$P\{\gamma(t+s) \leq \alpha \mid \gamma(s) = x, \gamma(u), 0 \leq u \leq s\} = P\{\gamma(t+s) \leq \alpha \mid \gamma(s) = x\}. \quad (2.1)$$

According to Singpurwalla [25], a disadvantage of the use of a Wiener process to model the diffusion of item state is that the sample path of wear is not monotonically increasing.

As an example of the failure rate (i.e. hazard rate function) being described by a stochastic process, Singpurwalla [25] considers a shot noise hazard rate process as follows. Suppose an item operates in a dynamic environment that inflicts shocks over time u according to a Poisson process with rate $m(u)$, $u \geq 0$. Each shock results in a stress on the item that contributes to its failure rate. If a shock of magnitude D occurs at an epoch S, and $h(t)$, a nonincreasing function of t, is the attenuation function, then the contribution of the shock to the item's failure rate is $Dh(t)$ at time $S + t$. The author presents the item's failure rate at time t as

$$\lambda(t) = \sum_{n=1}^{\infty} D_n h(t - S_n), \quad (2.2)$$

where $\{S_n, n \geq 1\}$ are the epochs at which the shocks occur, $\{D_n, n \geq 1\}$ are the respective shock magnitudes, and $h(t) = 0$ when $t < 0$.

In the case where the wear-inducing environment is modelled as a stochastic process, Singpurwalla [25] summarizes the use of Markov additive processes (MAP).

As an example of a MAP, one studies two stochastic processes: $X(t)$, which represents the cumulative damage of the item at time t, and $Z(t)$, which represents the state of the environment at time t. This MAP approach was used by Kharoufeh [16] and Kharoufeh and Sipe [17] and is reviewed in detail in the next chapter. Additionally, a wear-inducing shock process may be incorporated into this model. Singpurwalla [25] suggests that the MAP modelling approach, along with other stochastic modelling techniques, will become most useful only upon the further development of computational methodologies for reliability and survival analysis.

In the fourth and final strategy for the development of failure models studied in [25], the author briefly discusses the approach in which a response variable correlated with the lifelength is described by a stochastic process, such as a stationary, continuous-time Gaussian process. According to the author, there has been little development in this area of research. Overall, Singpurwalla [25] emphasizes the value of stochastic-based approaches for developing failure models, concluding that these models better exploit the physics of failure process. One drawback, as noted in [25], is the complexity of the resulting distributional forms, which must often be presented via Laplace transforms.

Abdel-Hameed [2] studies properties of the failure time distribution when the wear process is assumed to be an increasing Lévy process. Çinlar [10] introduced variations of a general shock and wear model where shocks of random magnitude occur at environment state transitions and cause degradation to the system. In [10], the author incorporates the wear and environment processes into a MAP and includes examples in which the wear process is both a Gamma and an increasing Lévy process. Esary et al. [12] consider the failure time distribution of a system subject to shocks governed by a Poisson process and then generalize the problem for cases of continuous wear. Limnios and Oprisan [19] apply Markov renewal theory to semi-Markov systems to obtain reliability and availability measures.

Kharoufeh [16] provides a brief overview of probabilistic techniques and notes that stochastic shock and wear models are the most commonly studied approaches. In the reliability analysis of a system that degrades over time, the author characterizes the continuous wear process as a nondecreasing stochastic process. In that work, the author considered a single-unit system whose cumulative damage over time is a continuous wear process dependent on an external environment process assumed to be a temporally homogeneous, Markov process with a finite state space. Kharoufeh [16] derived the failure time distribution and moments in terms of Laplace-Stieltjes transforms. Further discussion of the results in [16] is presented in the next chapter.

2.2 Statistical Approaches to Degradation-Based Reliability

In addition to the probabilistic models described above, the literature contains models for degradation-based failure distributions derived via nonlinear regression and other statistical techniques. Ahmad and Sheikh [3] present characteristics and applications of the Bernstein probability density function (pdf). Also known as the α-distribution, or inverted normal distribution, the Bernstein pdf was first developed by Gertsbakh and Kordonsky [14] and Vysokovskii [27] to model the life characteristics of machine components experiencing non-stationary linear wear. According to Ahmad and Sheikh [3], the Bernstein probability distribution has been successfully used to model cutting tool life, monitoring the dimensions of machine parts, testing the slideways and rotating parts of machine tools, and determining tool replacement intervals in precision machining. A potential drawback of the Bernstein probability distribution, with respect to its use in reliability analysis, is that it belongs to a bimodal family of probability distributions for which no moments exist [3]. This reliability model is based on a random wear process of the form

$$\mathbf{W}(t) = at + b, \tag{2.3}$$

where a is a random variable that describes the rate of wear, $\mathbf{W}(t)$, and $b = \mathbf{W}(0)$ is a random variable denoting the initial value of wear. When T is the lifetime of the component, and W_1 is the permissible wear limit, the three-parameter Bernstein's distribution is given by

$$F(t) = P(T \leq t) = \Phi\left[\frac{t-c}{\sqrt{\alpha t^2 + \beta}}\right] \tag{2.4}$$

where

$$\Phi(1/\sqrt{\alpha}) \simeq 1,$$

$$c = \frac{W_1 - E(b)}{E(a)},$$

$$\alpha = V(a)/E^2(a),$$

$$\beta = V(b)/E^2(a),$$

and

$$c, \alpha, \beta, E[(a)] > 0.$$

Ahmad and Sheik [3] further propose a two-parameter inverted family of distributions with $\beta = 0$ to model wear-related life. Their work includes a thorough study of the properties of the Bernstein distribution, including maximum likelihood estimates of its parameters. Lu and Meeker [20] expand the statistical study of degradation-based time-to-failure distributions by reviewing methods for fitting nonlinear regression models to observed fatigue-crack-growth measurements. The parameters of a mixed-effect path model are estimated using a two-stage method. The

authors present a parametric degradation model with fixed-effect parameters common for all units inspected and random-effect parameters, representing individual unit characteristics. Since expressing the time-to-failure distribution can be complicated by the inclusion of more than one random parameter, the authors proposed the use of Monte Carlo simulation to obtain the distribution numerically. For a degradation path modeled by

$$\eta(t) = \phi + \Theta(t), \tag{2.5}$$

where ϕ is fixed and represents the common initial amount of degradation of all test units, Θ, which represents the degradation rate, varies from unit to unit according to a Weibull(α, β) distribution, and D is the critical degradation level, the time-to-failure distribution is given by

$$F_T(t) = \exp\left[-\left(\frac{D-\phi}{\alpha t}\right)^{\beta}\right], t > 0. \tag{2.6}$$

Since the random variable 1/T follows a Weibull distribution, this time-to-failure distribution is known as the *reciprocal Weibull*. When Θ follows a lognormal distribution with parameters (μ, σ^2), T follows a lognormal distribution, and

$$F_T(t) = \Phi\left[\frac{log(t) - [log(D-\phi) - \mu]}{\sigma}\right]. \tag{2.7}$$

When $\Theta \sim N(\mu, \sigma^2)$ and $\sigma \ll \mu$ (to make $\mathrm{P}\{\Theta \leq 0\}$ negligible), the time-to-failure approximates a special case of the Bernstein distribution:

$$F_T(t) \approx \Phi\left[\frac{t - [D-\phi]/\mu}{\sigma t/\mu}\right]. \tag{2.8}$$

Lu and Meeker [20] present another form of the Bernstein distribution when both the intercept and the slope in the simple linear path model are assumed to be random, normally distributed, and independent of each other (i.e. ϕ is replaced by Θ_1, and

Θ is replaced by Θ_2 in (2.5).) In this case,

$$F_T(t) \approx \Phi\left(\frac{t - [D - \mu_1]/\mu_2}{\sqrt{[\sigma_1^2 + \sigma_2^2 t^2]/\mu_2^2}}\right). \tag{2.9}$$

Another nonlinear model presented in [20] is

$$\eta(t) = \phi_1 + \Theta \exp(\phi_2 t), \phi_2 > 0. \tag{2.10}$$

Here, ϕ_1 and ϕ_2 are fixed, and the resulting time-to-failure distribution is given as

$$F_T(t) \approx \Phi\left[\frac{t - [\log(D - \phi_1) - \mu]/\phi_2}{\sigma/\phi_2}\right]. \tag{2.11}$$

In this case,

$$T \sim N\left(\frac{\log(D - \phi_1) - \mu}{\phi_2}, \frac{\sigma^2}{\phi_2^2}\right). \tag{2.12}$$

Next, Lu and Meeker [20] present a multivariate normal model where the vector of random effects Θ follows a multivariate normal distribution with mean vector μ_θ and variance-covariance matrix Σ_θ. With this model, the time-to-failure distribution is expressed as

$$F_T(t) = F_T(t; \phi, \mu_\theta, \Sigma_\theta). \tag{2.13}$$

It is noted in [20] that there is generally no closed-form expression for this function. A two-stage method for estimating the random-effects parameters is proposed to avoid the computational intensity of full maximum likelihood estimation:

1. Fit the degradation model to the sample path for each sampled unit and obtain the Stage 1 estimates of the model parameters.

2. Model the random-effect parameters with a (multivariate) normal distribution by transforming (if necessary) the Stage 1 estimates.

3. Combine the transformed Stage 1 estimates of the model parameters to produce estimates of ϕ, μ_θ, and Σ_θ.

Once the estimated parameters are obtained, Lu and Meeker [20] propose a method by which Monte Carlo simulation is used to generate a sufficiently large number of random sample paths from the assumed path model (with the estimated parameters). The proportion failing is then used as a function of time as an estimate of $F_T(t)$. This method is particularly useful when there is no closed-form expression for $F_T(t)$ and when numerical transformation methods are too complicated. The authors present the following algorithm:

1. Estimate the path-model parameters ϕ, μ_θ, and Σ_θ;

2. Generate N simulated realizations $\tilde{\Theta}$ of Θ from $N(\mu_\Theta, \Sigma_\Theta)$ and obtain the corresponding N simulated realizations $\tilde{\Theta}$ of Θ;

3. Compute the corresponding N simulated failure times $\tilde{t}$ by substituting $\tilde{\Theta}$ into $T = \tau(\Theta; \hat{\phi}, D, \eta)$;

4. Estimate $F_T(t) = (\text{number of } \tilde{t} \leq t)/N$ for any desired values of t.

Lu and Meeker [20] then apply a parametric bootstrap method to obtain pointwise confidence intervals for $F_T(t)$. The same method is suggested for the construction of simultaneous confidence bands.

Chao [8] examines both fixed effect and random effect models. A simple fixed effect model for predicting the shelf life of drugs is given by

$$Y_{ij} = \alpha_i + \beta_i t_{ij} + \epsilon_{ij}, \tag{2.14}$$

where Y_{ij} is the result of the j-th assay of the i-th batch of drugs, t_{ij} is of dimension $p \times 1$ and denotes the appropriate regressor (time), α_i and β_i are fixed but unknown

parameters, and ϵ_{ij} denotes the error term. In this case, "fixed effect model" refers to the random selection of a single drug. Chao [8] notes that α_i and β_i can be modelled as random variables for random effect models, which refer to those models considering selection of a batch of drugs at random from many batches at hand. Chao [8] suggests that growth curve models provide a more general framework than the determination of shelf life. In cases where growth is bounded above, an S-shaped curve, or "sigmoid", may be used to model, from beginning to end, the changing growth rate. Chao [8] proposes the following model for growth:

$$y_i = \frac{\alpha}{1 + \exp[A - Bt_i]} + \epsilon_i, i = 1, 2, ..., n. \tag{2.15}$$

This model is often called a logistic growth model.

Chao [8] suggests there are two basic approaches used to describe a degradation process: the direct approach and the indirect approach. The direct approach begins with equations and uses data fitting for analysis. The indirect approach begins with theoretical justification and attempts to derive more theory without support from data. If $X(t)$ is a cumulative damage process with

$$X(t) = a + bt + W(t), \tag{2.16}$$

where $W(t)$ is a Brownian motion, then T has a distribution which is inverse Gaussian, and the exact solution can be found (cf. Bhattacharyya and Fries [5]).

Chinnam [9] developed a degradation-based approach for on-line drill-bit reliability determination through the use of neural networks for modeling degradation measures and self-organizing maps for modelling degradation variation. In [9] Chinnam proposes a failure time distribution based upon a nonlinear regressive degradation model with predictions of future degradation levels provided by the artificial neural network.

Crk [11] estimated degradation model parameters for every single unit, then combined them to obtain population parameters at every stress level. Crk [11] then uses multiple multivariate regression to extrapolate parameters at use stress levels. System reliability can be estimated using large numbers of generated failure times derived by generating large numbers of model parameters.

Wu and Tsai [28] considered the case where a few of the degradation paths in a test differ from the rest. After modelling the degradation paths via nonlinear regression, the authors used an optimal fuzzy clustering method to improve estimation of the random parameters and failure-time distribution.

Yacout, et al. [29] studied a mixed-effects degradation model for the case of cladding strain in irradiated fuel pins. Yacout, et.al. [29] used Monte Carlo simulation to obtain point estimates for the distribution and its confidence intervals of this mixed-effects model which is based on the work of Lu and Meeker [20].

Lawless [18] gave an overview of degradation models and failure-time distributions, including shared random effects models. The author's overview also includes Wiener and gamma processes and references the random growth curve in [20].

As this brief review of the literature attempts to show, many probabilistic and statistical methods for degradation-based reliability assessment are available and can be extremely useful to the practitioner. Although many statistical, degradation-based reliability techniques may provide accurate lifetime estimations, the use of techniques such as nonlinear regression requires observation and measurement of the system's degradation over time in order to construct a failure time model and estimate its parameters. Frequently, such measurements are prohibitively expensive or even impossible to obtain.

Probabilistic models in which the wear-inducing environment is stochastically modelled can also prove to be useful tools for reliability assessment. Unlike statistical, degradation-based models, a probabilistic approach does not require degradation

measurements during the lifetime of the item. However, restrictions on either the random environment or wear processes, such as the requirement for exponentially distributed state holding times seen in [16], limit the scope of reliability problems to be studied.

In this thesis, a probabilistic approach discussed by Singpurwalla [25] and thoroughly detailed by Kharoufeh [16] is chosen to evaluate the failure time distribution of a system under the influence of a dynamic environment. The objective is to develop a lifetime estimation model that does not rely upon degradation measurements during the system's lifetime and makes only minimal assumptions about the system's operating environment. Such a model provides an extremely useful tool for the reliability assessment of systems operating and degrading in dynamic environments. The next chapter utilizes and relaxes the assumptions of Kharoufeh [16] and Kharoufeh and Sipe [17] and provides the main results of this research effort.

3. Formal Mathematical Model

This chapter provides the mathematical model used to assess the failure time distribution and moments of failure time of a single-unit system experiencing nondecreasing wear due to a multi-state random environment process. Since it is assumed that the evolution of the environment depends only upon its current state, a discrete-time Markov chain (DTMC) underlies the environment process. Furthermore, since the distribution of random sojourn times is assumed to be nonexponential or even unknown, the environment must be characterized as a semi-Markov process (SMP). The model proposed herein converts the SMP environment process to a CTMC environment process, allowing for the results of Kharoufeh [16] and Kharoufeh and Sipe [17] to be applied.

In order for the environment to be characterized as a SMP, its true state space must first be partitioned into K distinct states. Such a partitioning may be achieved by observation and measurement or by consultation with subject matter experts. Having partitioned the environment, the rate at which each state of the environment causes degradation to the system must be obtained. In real-world applications, it is likely that subject matter experts can provide insight into the degradation rates associated with the states of a particular environment on a chosen system. In the proposed model, the degradation rate associated with each environment state is assumed to be a constant. Next, the environment process is observed over a long period of time. Holding times in each of the environment states, as well as the numbers of transitions between states, are recorded. Finally, the model is applied to obtain a reliability estimate of the system based on its failure time distribution and moments of failure time. A detailed discussion of this process follows in section 3.4.

Due to their relevance to this thesis research, the main results of [16] are reviewed here. Kharoufeh [16] derived the failure time distribution and moments of a single-unit system whose cumulative damage over time is a continuous wear process,

$\{X(t) : t \geq 0\}$, dependent on an external environment process. The environment process was assumed to be a temporally homogeneous, Markov process, $\{Z(t) : t \geq 0\}$ on a a finite state space $S = \{1, ..., K\}$. Kharoufeh [16] derived the cumulative distribution function of T_x, the time until failure, by using transform methods to solve a first-order linear partial differential equation satisfied by the joint probability distribution of the Markov additive process $\{(X(t), Z(t)) : t \geq 0\}$, conditioned upon the initial state of the environment. Using the derived failure time distribution, Kharoufeh [16] presented the Laplace-Stieltjes transform of the failure time moments. Define,

$$G(x,t) := P\{T_x \leq t\} \tag{3.1}$$

as the unconditional distribution of T_x, and the $K \times K$ matrix

$$\mathbf{V}(x,t) = [V_{i,j}(x,t)], \tag{3.2}$$

where

$$V_{i,j}(x,t) = P\{X(t) \leq x, Z(t) = j \mid Z(0) = i\} \tag{3.3}$$

is the joint probability that, at time t, the degradation of the system has not exceeded a value x, and the environment process is in state $j \in S$ given that the environment was initially in state $i \in S$. When $r : S \to \mathbb{R}^+$ is defined as a nonnegative function, the transform of the joint probability above is given as

$$\tilde{\mathbf{V}}^*(u,s) = [u\mathbf{R}_D + s\mathbf{I} - \mathbf{Q}]^{-1}, \tag{3.4}$$

where $Re(u) > 0, Re(s) > 0, \mathbf{R}_D = diag(r(1), ..., r(K))$, and $\mathbf{Q} = [q_{ij}]$ is the infinitesimal generator matrix of the Markov process, $\{Z(t) : t \geq 0\}$. The failure time

distribution of a single-unit system in a Markovian environment is given by

$$\tilde{G}^*(u,s) = \frac{1}{s} - \alpha\tilde{\mathbf{V}}^*(u,s)\mathbf{1}, Re(u) > 0, Re(s) > 0, \tag{3.5}$$

where $\alpha := [\alpha_i]$ is the initial distribution vector with $\alpha_i := P\{Z(0) = i\}$, and $\mathbf{1}$ is a K-dimensional column vector of ones. Kharoufeh and Sipe [17] simplified this expression to a one-dimensional Laplace-Stieltjes transform with respect to t, and their main result for the unit's failure time is given by

$$\tilde{G}_x(s) = \alpha \exp(\mathbf{R}_D^{-1}(\mathbf{Q} - s\mathbf{I})x)\mathbf{1}. \tag{3.6}$$

Let $\xi^n(x) := E[T_x^n]$ be the unconditional nth moment of the failure time. Then, if Z has initial probability distribution α, the Laplace-Stieltjes transform of the unconditional nth moment of the failure time is given by

$$\xi^n(u) = n!\alpha(u\mathbf{R}_D - \mathbf{Q})^{-n}\mathbf{1}. \tag{3.7}$$

Any one of several one-dimensional numerical Laplace transform inversion algorithms can be used to invert this transform. In [16] and [17], the authors provide a simplified method for estimating the reliability of a system based solely on the degradation effects of the operating environment on the system, where the environment process is assumed to be characterized as a CTMC.

The reliability model of Equation (3.6) strictly assumes that the holding times in each environment state are exponentially distributed random variables. This requirement facilitates the analysis of the wear process via a CTMC environment process. However, the real operating environment of a unit may not necessarily possess exponentially distributed state holding times. This thesis proposes a method that permits the relaxation of the requirement for exponentially distributed holding times while still applying the distribution result of Equation (3.6). In fact, the pro-

posed technique does not require knowledge of the distributions of the state holding times–only real observations of the sojourn times in each state. Furthermore, since arbitrary distributions can be represented exactly or approximately by phase-type distributions, such approximations will serve as the key to the relaxation of the requirement for exponentially distributed state holding times [23]. By approximating nonexponential state holding times with phase-type distributions, the Markovian property may be retained, thereby permitting analysis via CTMC-based techniques. In the following subsections, the rudimentary concepts of phase-type distributions are reviewed, including a few of their applications and techniques for obtaining their representations.

3.1 Phase-Type Distributions

In general, a phase-type distribution can be described as some mixture of exponential distributions, each representing a phase, with or without the same rate parameter. The phase-type random variable, described by the total time spent traversing the exponential phases, is equivalent to the time to absorption of an underlying Markov process. For example, a k-Erlang random variable, the sum of k independent and identically distributed exponential random variables, is equivalent to the absorption time of an underlying k-state Markov process. As noted by Perros [23], the value of phase-type distributions lies both in their possessing the Markovian (memoryless) property and their usefulness in either exactly or approximately representing arbitrary distributions. These two characteristics form the basis for the use of phase-type distributions in this work.

The use of phase-type distributions to model nonexponential holding times is thoroughly studied in the literature. One useful application of phase-type distributions is in approximating channel holding time distributions in mobile communications networks. Jayasuriya, et al [15] used generalized Erlang distributions to model channel holding time distributions, while Ro and Trivedi [24] used three-phase hy-

poexponential distributions. Boucherie and Van Dijk [6] apply mixtures of Erlang distributions as approximations to call length distributions and the distribution of the time spent in a cell. Fang and Chlamtac [13] utilized Erlang and hyper-Erlang call holding time distributions in their analysis of handoff probability. Phase-type distributions have also been used to approximate service times at different stations in a production line. Vidalis and Papadopoulos [26] studied the transition matrices of production lines by assuming a two-phase Coxian distribution for service times, where the first phase describes the service period, and the second phase describes the repair period which is reached with some specified probability.

The following is a slight modification of the description of the representation of a phase-type distribution as found in [23]. Recall that the total time spent traversing the k phases of a phase-type distribution is equivalent to the time to absorption of some underlying Markov process. Any phase-type distribution can be represented by a triplet $(\boldsymbol{\beta}, \mathbf{T}, \mathbf{T}^o)$, where $\boldsymbol{\beta} = (\beta, 0)$ is the $1 \times (k+1)$ initial probability vector of the overall phase space of the underlying Markov process, β is the $1 \times k$ vector giving the probabilities that the underlying Markov process will start in phase $i, i = 1, 2, ..., k$, $\mathbf{T}$ is the $k \times k$ matrix of transition rates among the first k phases, and $\mathbf{T}^o$ is the $k \times 1$ vector of transition rates out of the first k phases into the absorbing phase $k+1$. Note that $\boldsymbol{\beta} = (\beta, 0)$ implies that the underlying Markov process never begins in the absorbing phase $k+1$. The rate matrix of the phase-type distribution (and the underlying Markov process) is

$$\mathbf{S} = \begin{pmatrix} \mathbf{T} & \mathbf{T}^o \\ \mathbf{0} & 1 \end{pmatrix}. \tag{3.8}$$

Using the representation $(\boldsymbol{\beta}, \mathbf{T}, \mathbf{T}^o)$, the density function of a phase-type distribution is given by

$$f(x) = \beta \exp(\mathbf{T}x)\mathbf{T}^o, x \geq 0, \tag{3.9}$$

with Laplace transform

$$f^*(s) = \beta(s\mathbf{I} - \mathbf{T})^{-1}\mathbf{T}^o, Re(s) \geq 0. \tag{3.10}$$

The nth moment ($n \geq 1$) of a phase-type distribution with representation $(\boldsymbol{\beta}, \mathbf{T})$ is given by

$$E[X^n] = (-1)^n n!\beta\mathbf{T}^{-n}\mathbf{1}. \tag{3.11}$$

The next two sections describe various phase-type distributions, their structures of their representations, and techniques for obtaining the parameters of their representations.

3.2 Types of Phase-Type Distributions

In this subsection, various types of phase-type approximations are briefly examined. In the next section, the technique used herein to construct phase-type approximations when state holding times are known, either via observed data or known parametric probability distributions, is reviewed.

3.2.1 General Phase-Type Distribution

The general case of a phase-type distribution, denoted by PH_k, is one in which the exponential phases are not necessarily visited sequentially. In other words, a visit to any phase $i, i = 1, ..., k$, can be followed by a visit to any other phase $j, j = 1, ...k, i \neq j$, with probability a_{ij} prior to absorption. It is assumed here that absorption occurs upon reaching an unseen absorbing phase. Furthermore, each phase $i, i = 1, 2, ..., k$, has exponential rate parameter μ_i. Figure 3.1 gives a graphical depiction of the general case of a phase-type distribution. For the sake of clarity, some of the transition probabilities have been omitted.

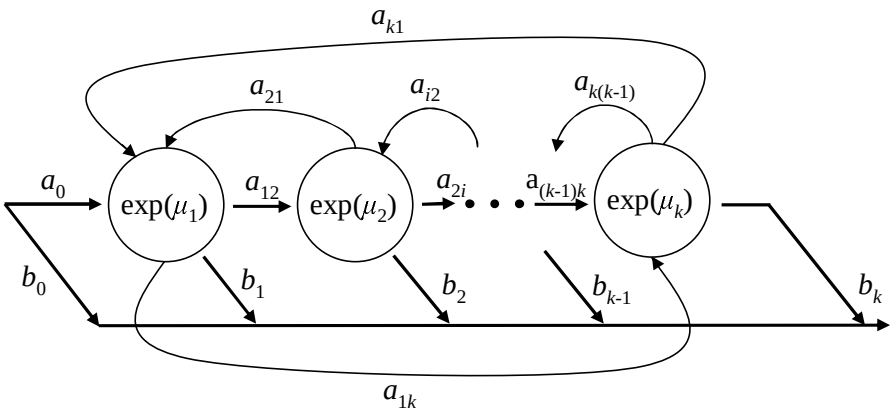

Figure 3.1 Graphical depiction of a k-phase, phase-type distribution.

3.2.2 Coxian Distribution

The k-phase Coxian distribution, denoted by C_k, consists of a sequence of k exponential distributions with respective rate parameters $\mu_i, i = 1, 2, ..., k$. As a special case of the general phase-type distribution, the Coxian distribution is sequential in its phases and does not permit transitions from phase j to phase i when $j > i$.

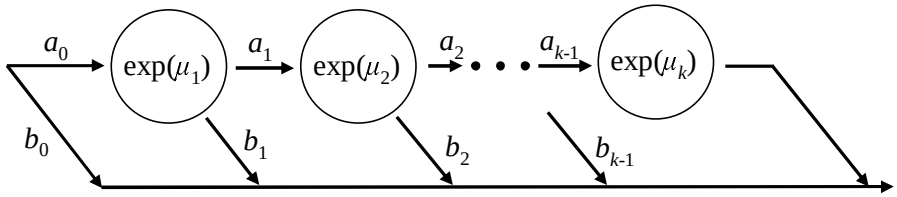

Figure 3.2 Graphical depiction of a k-phase Coxian distribution.

In Figure 3.2, $a_i, i = 0, 1, ..., k-1$, denotes the probability of departing phase i and entering phase $i+1$, while $b_i, i = 0, 1, ..., k-1$, denotes the probability of departing phase i and entering the absorbing phase. Also, note that a_0 denotes the probability of entering phase 1, while b_0 denotes the probability of zero service time. A useful property of the Coxian distribution is that it can exactly represent any distribution having a rational Laplace transform [23]. Moreover, Perros [23] gives the Laplace

transform of a Coxian distribution as

$$f^*(s) = b_0 + \sum_{i=1}^{k} a_0 ... a_{i-1} b_i \prod_{j=1}^{i} \frac{\mu_j}{s + \mu_j} \tag{3.12}$$

where $a_i + b_i = 1, i = 1, 2, ..., k-1$, and $b_k = 1$.

3.2.3 Erlang Distribution

The Erlang distribution is a special case of the Coxian in which each exponential phase has equal rate parameter μ, and there is zero probability of entering the absorbing phase (or any phase other than phase $i+1$) until the kth phase is reached [23]. Figure 3.3 gives a graphical depiction of this distribution.

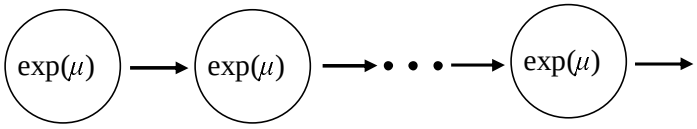

Figure 3.3 Graphical depiction of a k-phase Erlang distribution with rate parameter μ.

A more general form of the Erlang distribution, depicted in Figure 3.4, is obtained by allowing distinct exponential phase rate parameters $\mu_i, i = 1, 2, ..., k$.

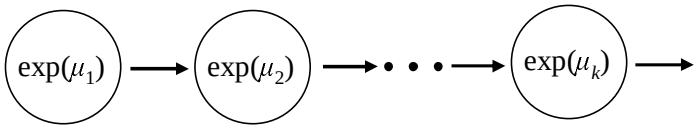

Figure 3.4 Graphical depiction of a k-phase Erlang distribution with distinct rate parameters.

The generalized Erlang distribution is a k-phase Erlang distribution that transitions from phase 1 to phase 2 with probability a and from phase 1 to the absorbing phase with probability 1-a. If phase 2 is reached, then all phases after phase 2 are also reached (sequentially). Note that an alternative definition of the generalized Erlang

distribution, equivalent to the multiple rate Erlang distribution described above, is implied in [21].

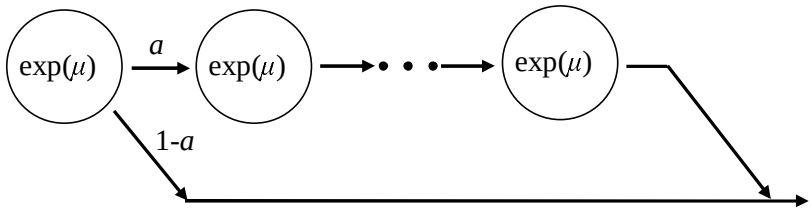

Figure 3.5 Graphical depiction of a k-phase generalized Erlang distribution with single rate parameter.

Since the time required to traverse all phases of a k-phase Erlang distribution is the sum of k independent and identically distributed, exponential random variables with common rate parameter μ, the Laplace transform (LT) of the k-phase Erlang distribution is given by

$$f^*(s) = \left(\frac{\mu}{s+\mu}\right)^k, \tag{3.13}$$

and the LT of the k-phase Erlang distribution with distinct rate parameters $\mu_i, i = 1, 2, ..., k$, is

$$f^*(s) = \prod_{i=1}^{k} \left(\frac{\mu_i}{s+\mu_i}\right). \tag{3.14}$$

3.3 Phase-type Approximation Technique

In this section, the procedure used in this thesis to approximate observed data or parametric probability distributions with phase-type distributions is presented. Table 3.1 provides some guidelines for choosing the particular phase-type approximation to use based on the squared coefficient of variation of the observed data or parametric probability distribution [23]. More specifically, suppose X is an arbitrary

nonnegative random variable. The objective is to obtain a phase-type distribution to approximate the true distribution of X. The squared coefficient of variation for the random variable X is

$$c^2 = \frac{Var[X]}{(E[X])^2}. \tag{3.15}$$

Table 3.1 Guidelines for selection of phase-type approximation.

Range of c^2	Distribution
$c^2 > 1$	three-moment, 2-phase Coxian
$0.5 \leq c^2 \leq 1$	two-moment, 2-phase Coxian
$c^2 < 0.5$	two-moment, k-phase generalized Erlang

The following calculations follow from [4] and [23].

3.3.1 Approximation via three-moment, 2-phase Coxian distribution

When $c^2 > 1$, the observed holding time data or parametric probability distribution is approximated with a three-moment, 2-phase Coxian distribution. The Laplace transform (LT) of this approximation is given as

$$f^*(s) = \frac{\mu_1 s(1-a) + \mu_1\mu_2}{s^2 + (\mu_1 + \mu_2)s + \mu_1\mu_2}, \tag{3.16}$$

where s is a complex transform variable, and μ_1, μ_2, and a denote the parameters of the Coxian representation as described in section 3.2. After taking successive derivatives of the above LT and evaluating at $s = 0$, the first three moments of the C_2 distribution may be obtained. Let Y be a 2-phase Coxian distributed random variable. Estimates (or exact values) of the first three moments (m_1, m_2, and m_3, respectively) of the observed data or parametric probability distribution are set equal to the three C_2 moments and are expressed as (cf. [4])

$$m_1 \equiv E[Y] = \frac{1}{\mu_1} + \frac{a}{\mu_2}, \tag{3.17}$$

$$m_2 \equiv E[Y^2] = \frac{2(1-a)}{\mu_1^2} - \frac{2a\mu_1\mu_2 - 2a(\mu_1+\mu_2)^2}{\mu_1^2\mu_2^2}, \tag{3.18}$$

and

$$m_3 \equiv E[Y^3] = \frac{6(1-a)}{\mu_1^3} - \frac{12a\mu_1\mu_2(\mu_1+\mu_2) - 6a(\mu_1+\mu_2)^3}{\mu_1^3\mu_2^3}. \tag{3.19}$$

For convenience, the following equalities are introduced:

$$A = \mu_1 + \mu_2 \tag{3.20}$$

and

$$B = \mu_1\mu_2. \tag{3.21}$$

Upon substitution and simplification of Equations (3.17), (3.18), and (3.19), the following expressions are obtained:

$$A = \frac{1}{m_1} + \frac{m_2 B}{2m_1} \tag{3.22}$$

and

$$B = \frac{12m_1^2 - 6m_2}{3m_2^2 - 2m_1m_3}. \tag{3.23}$$

Following [4], the parameters of the three-moment, 2-phase Coxian distribution are

$$\mu_1 = A + \frac{\sqrt{A^2 - 4B}}{2}, \tag{3.24}$$

$$\mu_2 = A - \mu_1, \tag{3.25}$$

and

$$a = \frac{\mu_2}{\mu_1}(m_1\mu_1 - 1). \tag{3.26}$$

The three-moment, 2-phase Coxian distribution has the representation

$$\mathbf{T} = \begin{pmatrix} -\mu_1 & a\mu_1 \\ 0 & -\mu_2 \end{pmatrix}, \tag{3.27}$$

$$\boldsymbol{\beta} = (1, 0, 0), \tag{3.28}$$

$$\mathbf{T^o} = [(1-a)\mu_1, \mu_2]. \tag{3.29}$$

3.3.2 Approximation via two-moment, 2-phase Coxian distribution

When $0.5 \leq c^2 \leq 1$, a two-moment, 2-phase Coxian dsitribution is used to approximate the observed holding time data or parametric probability distribution. Following a moment-matching algorithm similar to that for the three-moment 2-phase Coxian distribution, the parameters of the two-moment variant of the 2-phase Coxian are

$$\mu_1 = \frac{2}{m_1}, \tag{3.30}$$

$$\mu_2 = \frac{1}{m_1 c^2}, \tag{3.31}$$

and

$$a = \frac{1}{2c^2}. \tag{3.32}$$

The two-moment, 2-phase Coxian distribution has the same representation as the three-moment, 2-phase Coxian distribution described above. It is important to note that using either variant of the 2-phase Coxian distribution as an approximation to

observed holding time data or a parametric probability distribution helps to minimize the growth of the overall state space. The drawback of state space growth is further discussed in section 3.4.1.

3.3.3 Approximation via two-moment, k-phase Erlang distribution

When the squared coefficient of variation is less than 0.5, the observed holding time data or parametric probability distribution is approximated by a two-moment, k-phase generalized Erlang approximation. Following [23], the number of phases k is chosen such that

$$\frac{1}{k} \leq c^2 \leq \frac{1}{(k-1)}. \tag{3.33}$$

In the numerical examples of chapter 4, a simple linear program is implemented to approximate the minimum number of phases k. The remaining parameters, μ and a, corresponding to those shown in Figure 3.5, are computed as

$$1 - a = \frac{2kc^2 + k - 2 - (k^2 + 4 - 4kc^2)^{1/2}}{2(c^2+1)(k-1)} \tag{3.34}$$

and

$$\mu = \frac{1 + (k-1)a}{m_1}. \tag{3.35}$$

The k-phase generalized Erlang distribution is then represented by

$$\mathbf{T} = \begin{pmatrix} -\mu & a\mu & 0 & 0 & \cdots & 0 \\ 0 & -\mu & \mu & 0 & \cdots & 0 \\ 0 & 0 & -\mu & \mu & \ddots & \vdots \\ 0 & 0 & 0 & -\mu & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \mu \\ 0 & 0 & 0 & \dots & 0 & -\mu \end{pmatrix} \tag{3.36}$$

$$\boldsymbol{\beta} = (1, 0, \cdots, 0) \tag{3.37}$$

$$\mathbf{T^o} = [(1-a)\mu, 0, \cdots, 0, \mu]. \tag{3.38}$$

3.4 Phase-Type Approximations for Degradation-Based Reliability

In real-world situations, when analyzing the wear process of a single-unit system subject to a multi-state stochastic environment process, knowledge of the environment process will likely be limited to the degradation rate associated with each state and observations of the state holding times. Since there is no guarantee that these holding times will be exponentially distributed, the environment process cannot be assumed to be a continuous-time Markov chain. However, phase-type distributions, such as those described in section 3.2, may be used as approximations to these nonexponential distributions to impose the Markovian property. That is, the approximated environment process may be modelled as a continuous-time Markov chain (CTMC), and subsequently a new (and expanded) infinitesimal generator matrix may be used directly in Equation (3.6).

3.4.1 Conversion of the Environment Process to a CTMC

Consider a single-unit system subject to wear over time due to a multi-state semi-Markov environment process, $\{Z(t) : t \geq 0\}$, with finite state space $S = \{1, 2, \ldots, K\}$. For the sake of brevity, the SMP is abbreviated as Z. The objective is to convert Z into a CTMC. To accomplish this, each sojourn time distribution $H_i, i \in S$, of Z is approximated by a phase-type distribution $\hat{H}_i$. The state space $\hat{S}$ of the resulting CTMC consists of all of the phases in the K phase-type approximations H_i.

Initially, the environment process must be observed over a long time interval of length τ. The holding times in the K states and the random number of transitions from one state to another are observed and recorded. The observed rate of transition from state i to state j, $i, j \in S$, over τ is computed and denoted as $\hat{q}_{ij}$. The $K \times K$

transition rate matrix $\mathbf{Q}$, whose elements are $\hat{q}_{ij}$, is then formed. After computing the required moments of the observed holding times in each state, the K holding time distributions (H_i, $i = 1, 2, ..., K$) are approximated by K phase-type distributions ($\hat{H}_i$, $i = 1, 2, ..., K$) by applying the techniques of section 3.3. Recall that either a 2-phase Coxian or a k-phase generalized Erlang distribution is chosen to approximate each holding time distribution depending on the estimated squared coefficient of variation of each state's observed holding times.

A potential drawback of this technique is the possibility of a state space explosion. For example, consider a five-state semi-Markov environment process with arbitrary sojourn time distributions $H_i, i = 1, 2, ..., 5$. If each distribution is approximated by a four-phase, generalized Erlang distribution, the resulting CTMC will have twenty states, represented by the phases used to approximate each distribution. In an effort to minimize the dimensionality of the new state space, and therefore ease the computational burden, one should use techniques that result in a minimal number of phases for each sojourn time distribution. The recent work of Osogami and Harchol-Balter [22] aims at minimizing the number of phases while maximizing the accuracy of the phase-type approximation.

Define k_i as the number of phases used to approximate sojourn time distribution H_i. For any state $i \in S$, the underlying Markov process has $k_i + 1$ phases, including k_i exponential phases plus one absorbing phase. In order to complete the conversion of Z into a CTMC, the infinitesimal generator matrix $\mathbf{\Psi}$, whose elements represent the transition rates among all phases of the newly partitioned K states, must be created. Since the initial probability vector $\boldsymbol{\beta}_i$ of each phase-type approximation is of the form $(1, 0, \ldots, 0)$, transitions from state i to state j for $i \neq j$ are limited to transitions from the absorbing phase of state i to the first phase of state j. In other words, any state must be exited via its absorbing phase, and a new state must be entered via its first phase.

As an example, suppose one seeks to convert (approximately) a K-state semi-Markov environment process into a CTMC using 2-phase Coxian distributions for each of the K states. After the number of transitions among the K states of the original environment process have been observed and recorded, the observed rates of transition from state $i, i = 1, 2, \ldots, K$, to state $j, j = 1, 2, \ldots, K$, are computed as

$$\hat{q}_{ij} = \frac{N_\tau(i,j)}{H_\tau(i)}, \tag{3.39}$$

where $N_\tau(i,j)$ is the total number of transitions from state i to state j in time τ, and $H_\tau(i)$ is the total time spent in state i during time τ. Figure 3.6 gives a graphical depiction of the approximation of such an environment when $K = 3$. This

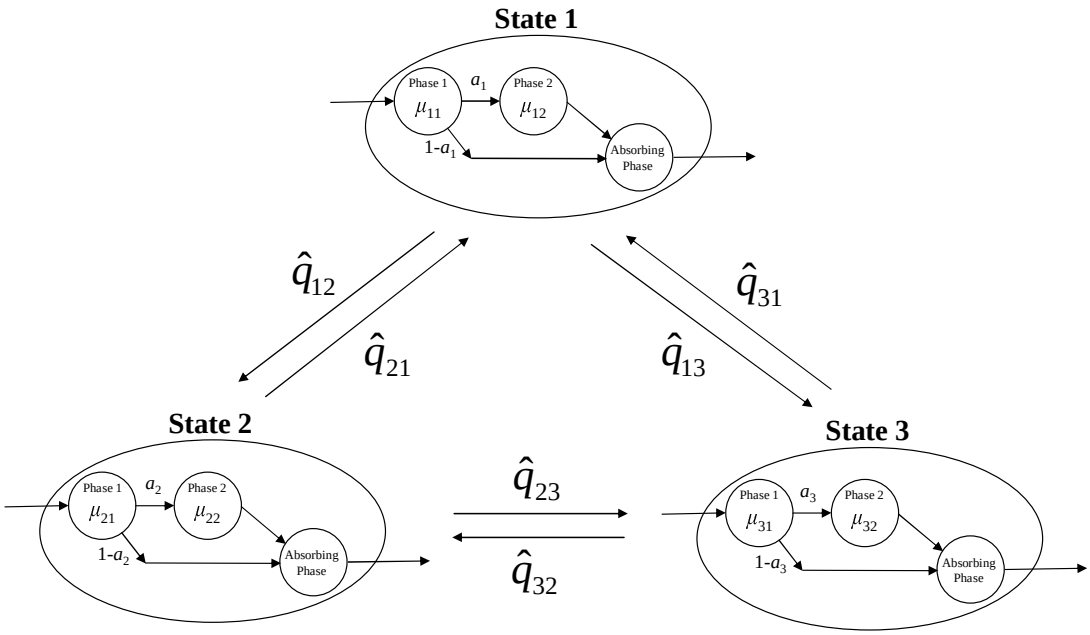

Figure 3.6 Graphical depiction of a 3-state environment process with 2-phase Coxian sojourn time approximations.

K-state CTMC environment process possesses an infinitesimal generator matrix of

the following form:

$$\mathbf{\Psi} = \begin{pmatrix} \mathbf{\Psi}_{11} & \mathbf{\Psi}_{12} & \dots & \mathbf{\Psi}_{1K} \\ \mathbf{\Psi}_{21} & \mathbf{\Psi}_{22} & \dots & \mathbf{\Psi}_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{\Psi}_{K1} & \mathbf{\Psi}_{K2} & \dots & \mathbf{\Psi}_{KK} \end{pmatrix}, \tag{3.40}$$

where

$$\mathbf{\Psi}_{ii} = \begin{pmatrix} \mathbf{T}_i & \mathbf{T}_i^o \\ \mathbf{0} & -\sum_{i \neq j} \hat{q}_{ij} M \end{pmatrix}, \quad i \in S, \tag{3.41}$$

$$\mathbf{\Psi}_{ij} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \hat{q}_{ij} M & 0 & 0 \end{pmatrix}, \quad i \neq j \in S, \tag{3.42}$$

and $\mathbf{T}_i$ and $\mathbf{T}_i^o$ are the phase-type representative $\mathbf{T}$ matrix and $\mathbf{T}^o$ vector, respectively, of state i (see section 3.1).

The elements of $\mathbf{\Psi}_{ii}, i \in S$, represent the rates of transition among the $k_i + 1$ phases of the newly partitioned state i, and the elements of $\mathbf{\Psi}_{ij}, i \neq j \in S$, represent the rates of transition from the $k_i + 1$ phases of state i to the $k_j + 1$ phases of state j. In other words, $\mathbf{\Psi}_{ii}, i \in S$, gives the transition rates among the $k_i + 1$ phases of a single state, and $\mathbf{\Psi}_{ij}, i \neq j \in S$, gives the transition rates from the $k_i + 1$ phases of one state to the $k_j + 1$ phases of another state. It is helpful to note that, in the general case where k-phase, phase-type distributions are used to approximate K environment states, the dimensions of $\mathbf{\Psi}_{ii}$, are $(k_i+1) \times (k_i+1)$, and the dimensions of $\mathbf{\Psi}_{ij}, i \neq j$, are $(k_i+1) \times (k_j+1)$. It is also important to note that the holding time in the absorbing phase of any state is theoretically instantaneous, thereby implying an infinite transition rate. Therefore, in numerical implementations, it is necessary that the transition rate from the absorbing phase of any state to either itself or the

first phase of any other state is multiplied by a very large number M. In other words, each $\hat{q}_{ij}$ in $\mathbf{\Psi}$ is multiplied by M to create a very large transition rate.

Given that the wear rate in environment state i is $r(i)$, $i = 1, 2, \ldots, K$, the wear rate matrix of the semi-Markov environment process is

$$\mathbf{R}_D = \begin{pmatrix} r(1) & 0 & \ldots & 0 \\ 0 & r(2) & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \ldots & 0 & r(K) \end{pmatrix}. \tag{3.43}$$

Having expanded the original generator matrix to account for transitions among the phases of the approximated environment states, the corresponding wear rate matrix $\mathbf{R}_D$ must also be expanded. The wear rate $r(i)$ of each environment state i, $i = 1, 2, \ldots, K$, is assumed to be the same for all $k_i + 1$ phases of the phase-type approximation of state i. Therefore, if each state is approximated via a 2-phase Coxian distribution, the expanded wear rate matrix of the K-state example is given as

$$\mathbf{\Lambda} = \begin{pmatrix} r(1) & 0 & 0 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 \\ 0 & r(1) & 0 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 \\ 0 & 0 & r(1) & 0 & 0 & 0 & \ldots & 0 & 0 & 0 \\ 0 & 0 & 0 & r(2) & 0 & 0 & \ldots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & r(2) & 0 & \ldots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & r(2) & \ddots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \ldots & 0 & r(K) & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \ldots & 0 & 0 & r(K) & 0 \\ 0 & 0 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & r(K) \end{pmatrix}. \tag{3.44}$$

In the next subsection, $\mathbf{\Psi}$ and $\mathbf{\Lambda}$ are used to compute the failure time distribution and moments of failure of the single-unit system under study.

3.4.2 Computation of Failure Time Distribution and Moments

Once the semi-Markov environment has been converted into a CTMC with infinitesimal generator matrix $\mathbf{\Psi}$ and wear rate matrix $\mathbf{\Lambda}$, the results derived by Kharoufeh [16] and Kharoufeh and Sipe [17] may be applied to compute the cumulative distribution function of the failure time of the single-unit system under study. Figure 3.7 provides a summary of the overall process. Applying Equations (3.6) and (3.7), the Laplace-Stieltjes transform of the failure time distribution is given by

$$\tilde{G}_x(s) = \alpha \exp(\mathbf{\Lambda}^{-1}(\mathbf{\Psi} - s\mathbf{I})x)\mathbf{1}, \tag{3.45}$$

and the Laplace-Stieltjes transform of the nth moment of unit failure time is given by

$$\xi^n(u) = n!\alpha(u\mathbf{\Lambda} - \mathbf{\Psi})^{-n}\mathbf{1}, \tag{3.46}$$

where α is the $1 \times K$ initial probability vector determining the beginning state of the environment process. Any one of several one-dimensional inversion algorithms may be used to numerically invert these transforms to obtain $G_x(t)$ and $\xi^n(x)$, the failure time distribution and nth moment of unit failure time, respectively. The next chapter provides three numerical examples that illustrate the conversion of a semi-Markov environment process to a CTMC environment process via phase-type approximations. The failure time distribution and lower moments of failure time are estimated for each system under consideration.

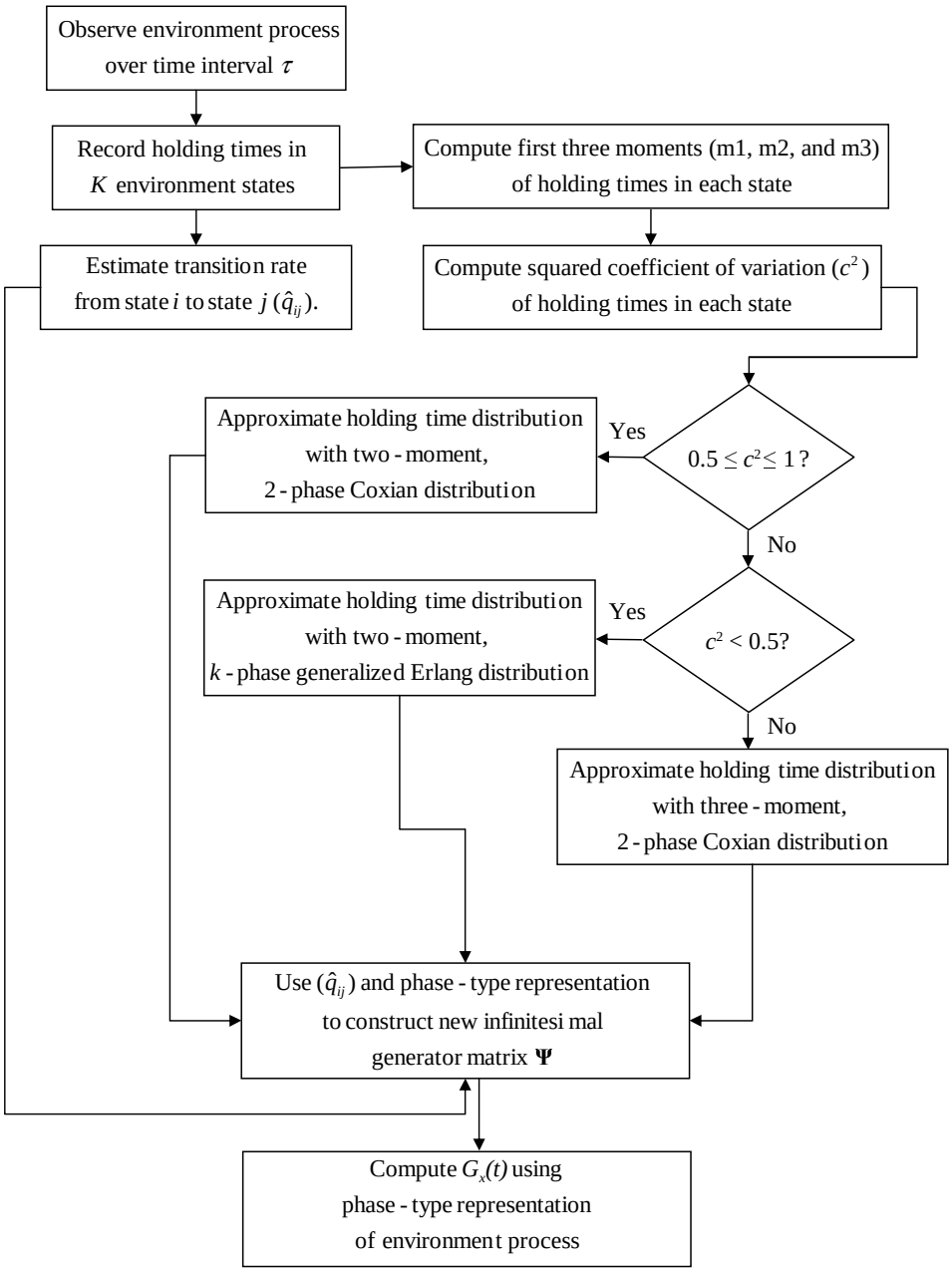

Figure 3.7 Flow diagram for reliability assessment in a SMP environment.

4. Numerical Results

In this chapter, applications of the reliability evaluation process described in chapter 3 are illustrated. Using phase-type approximation techniques, the failure time distributions for single-unit systems subject to semi-Markov environment processes will be evaluated via the results previously derived for systems subject to environment processes characterized as CTMCs. Additionally, the first and second moments of failure time will also be evaluated. The results are directly applicable to real-world situations.

4.1 Degradation of Brake Pad Material

Consider a new brake pad installed on an automobile equipped with disc brakes. In a disc brake system, a caliper holds the brake pad and pushes it against a rotor on the automobile's wheel when the brake pedal is depressed. In this example, assume that the brake pad is always in contact with the rotor. (In modern disc brake systems, the brake pad sits clear of the rotor when the brakes are not applied.) When the brakes are applied, the caliper pushes the brake pad against the rotor with a force proportional to the pressure applied to the brake pedal. The friction generated between the brake pad and the rotor causes the automobile to slow down. Over time, the brake pad experiences normal wear due to the friction between the pad and the rotor and eventually reaches a failure threshold x. Reaching this threshold may not necessarily lead to system failure but may be used to indicate the need for preventive maintenance of the braking system. The degradation rate of the brake pad is linear and completely determined by the system's random environment, which consists of three states. State 1, which is described by the normal contact of the brake pad with the rotor, causes the brake pad to wear with rate $r(1)$. States 2 and 3, achieved by depressing the brake pedal with two distinct pressures, cause the brake pad to wear with rates $r(2)$ and $r(3)$, respectively. It is assumed that the

system's random environment is characterized by a semi-Markov process with state space $S = \{1, 2, 3\}$. Let $\{X_n : n \geq 0\}$ be a DTMC embedded within the SMP such that X_n is the state of the environment just after the nth transition. The DTMC has transition probability matrix

$$\mathbf{P} = \begin{pmatrix} 0 & a & 1-a \\ b & 0 & 1-b \\ c & 1-c & 0 \end{pmatrix},$$

where a is the probability of transitioning from state 1 to state 2, b is the probability of transitioning from state 2 to state 1, and c is the probability of transitioning from state 3 to state 1.

Let $X(t)$ denote the cumulative degradation of the brake pad at time t, let $\mathbf{R}_D$ be the diagonal matrix of wear rates $r(1)$, $r(2)$, and $r(3)$, and let T_x denote the random time required for the brake pad's cumulative degradation to reach amount x. The cumulative distribution function of T_x is desired.

In order to apply the results of [16] for the failure-time distribution, the semi-Markov environment process is first converted into a CTMC via phase-type approximations of the state holding time distributions. After observing the random environment over time τ, the transition rates $\hat{q}_{ij}$, $i, j \in S$, are computed for the transitions among the environment's three states. This is accomplished by dividing the number of times the system transitions from state i to state j, $i \neq j$, by the total amount of time the system spends in state i for $i, j \in S$. Additionally, the observed holding times in each state of the environment are recorded and used to construct phase-type approximations to the holding time distributions. The resulting representations of the phase-type distributions and the observed state transition rates are used to construct $\mathbf{\Psi}$, the generator matrix of the resulting CTMC. Using the generator matrix, $\mathbf{\Psi}$, the expanded degradation rate matrix, $\mathbf{\Lambda}$, and the initial

probability distribution of the environment states, α, as input, Equation (3.45) is applied to calculate the solution of the failure time distribution in the transform.

For this example, the wear rates are $r(1) = 0.1$, $r(2) = 1.0$, and $r(3) = 2.0$. In order to conduct a simulation of the environment process, the state holding time distributions are assumed to be known and are given in Table 4.1. Additionally,

Table 4.1 State holding time distributions for the brake pad example.

State	Distribution
1	Weibull(2,4)
2	Weibull(3,5)
3	Weibull(4,6)

the transition probability matrix values are $a = 0.7, b = 0.6$, and $c = 0.2$, and the brake pad damage threshold is set at $x = 30.0$. Furthermore, it is assumed that the environment process always begins in State 1. Therefore, the required input matrices are

$$\mathbf{P} = \begin{pmatrix} 0.0 & 0.7 & 0.3 \\ 0.6 & 0.0 & 0.4 \\ 0.2 & 0.8 & 0.0 \end{pmatrix},$$

$$\mathbf{R}_D = \begin{pmatrix} 0.1 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 2.0 \end{pmatrix},$$

and

$$\alpha = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}.$$

After observing the system for $\tau = 10000$ time units, the $\mathbf{T}_i$ matrices and $\mathbf{T}_i^o$ vectors of the phase-type representations of the three holding time distributions were

estimated as

$$\mathbf{T}_1 = \begin{pmatrix} -8.826 & 8.562 & 0.000 & 0.000 \\ 0.000 & -8.826 & 8.826 & 0.000 \\ 0.000 & 0.000 & -8.826 & 8.826 \\ 0.000 & 0.000 & 0.000 & -8.826 \end{pmatrix}, \quad \mathbf{T}_1^o = \begin{pmatrix} 0.2633 \\ 0 \\ 0 \\ 8.8256 \end{pmatrix},$$

$$\mathbf{T}_2 = \begin{pmatrix} -15.2250 & 15.1090 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -15.2250 & 15.2250 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -15.2250 & 15.2250 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -15.2250 & 15.2250 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -15.2250 & 15.2250 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -15.2250 & 15.2250 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -15.2250 & 15.2250 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -15.2250 \end{pmatrix},$$

$$\mathbf{T}_2^o = \begin{pmatrix} 0.1135 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 15.2290 \end{pmatrix},$$

$$\mathbf{T}_3 = [\delta_{ij}] \quad \text{for} \quad i = 1, \ldots, 13, j = 1, \ldots, 13,$$

where

$$\delta_{ij} = \begin{cases} -22.4030, & i = j \\ 22.3580, & i = 1,\ j = 2 \\ 22.4030, & i = j - 1,\ j = 3, \ldots, 13 \\ 0 & \text{otherwise} \end{cases},$$

and

$$\mathbf{T}_3^o = [\gamma_i] \quad \text{for} \quad i = 1, \ldots, 13,$$

where

$$\gamma_i = \begin{cases} 0.0450, & i = 1 \\ 0, & i = 2, \ldots, 12 \\ 22.4030, & i = 13 \end{cases} .$$

Furthermore, the observed transition rates among the three environment states are presented as the elements of $\hat{\mathbf{Q}}$, where

$$\hat{\mathbf{Q}} = \begin{pmatrix} -2.2569 & 1.5809 & 0.6760 \\ 1.1501 & -1.9162 & 0.7661 \\ 0.3403 & 1.3862 & -1.7265 \end{pmatrix} .$$

Using the above phase-type representations and the $\hat{q}_{ij}$ elements of $\hat{\mathbf{Q}}$, the 28×28 generator matrix of the newly formed CTMC is

$$\mathbf{\Psi} = \begin{pmatrix} \mathbf{\Psi}_{11} & \mathbf{\Psi}_{12} & \mathbf{\Psi}_{13} \\ \mathbf{\Psi}_{21} & \mathbf{\Psi}_{22} & \mathbf{\Psi}_{23} \\ \mathbf{\Psi}_{31} & \mathbf{\Psi}_{32} & \mathbf{\Psi}_{33} \end{pmatrix},$$

where

$$\mathbf{\Psi}_{11} = \begin{pmatrix} \mathbf{T}_1 & \mathbf{T}_1^o \\ \mathbf{0} & -22569.0 \end{pmatrix},$$

$$\mathbf{\Psi}_{12} = [\psi_{ij}] \text{ for } i = 1, \ldots, 5, j = 1, \ldots, 9,$$

where

$$\psi_{ij} = \begin{cases} 15809.0, & i = 5,\ j = 1 \\ 0 & \text{otherwise} \end{cases},$$

$$\mathbf{\Psi}_{13} = [\psi_{ij}] \text{ for } i = 1, \ldots, 5, j = 1, \ldots, 14,$$

where

$$\psi_{ij} = \begin{cases} 6760.2, & i = 5,\ j = 1 \\ 0 & \text{otherwise} \end{cases},$$

$$\mathbf{\Psi}_{21} = [\psi_{ij}] \text{ for } i = 1, \ldots, 9, j = 1, \ldots, 5,$$

where

$$\psi_{ij} = \begin{cases} 11501.0, & i = 9,\ j = 1 \\ 0 & \text{otherwise} \end{cases},$$

$$\mathbf{\Psi}_{22} = \begin{pmatrix} \mathbf{T}_2 & \mathbf{T}_2^o \\ \mathbf{0} & -19162.0 \end{pmatrix},$$

$$\mathbf{\Psi}_{23} = [\psi_{ij}] \text{ for } i = 1, \ldots, 9, j = 1, \ldots, 14,$$

where

$$\psi_{ij} = \begin{cases} 7660.9, & i = 9,\ j = 1 \\ 0 & \text{otherwise} \end{cases},$$

$$\mathbf{\Psi}_{31} = [\psi_{ij}] \text{ for } i = 1, \ldots, 14, j = 1, \ldots, 5,$$

where

$$\psi_{ij} = \begin{cases} 3403.1, & i = 14,\ j = 1 \\ 0 & \text{otherwise} \end{cases},$$

$$\mathbf{\Psi}_{32} = [\psi_{ij}] \text{ for } i = 1, \ldots, 14, j = 1, \ldots, 9,$$

where

$$\psi_{ij} = \begin{cases} 13862.0, & i = 14,\ j = 1 \\ 0 & \text{otherwise} \end{cases},$$

and

$$\mathbf{\Psi}_{33} = \begin{pmatrix} \mathbf{T}_3 & \mathbf{T}_3^o \\ \mathbf{0} & -17265.0 \end{pmatrix}.$$

As noted in chapter 3, the holding time in the absorbing phase of any state is theoretically instantaneous, thereby implying a theoretically infinite transition rate. Therefore, the transition rate from the absorbing phase of any state to either itself or the first phase of any other state is multiplied by a large number M when constructing the infinitesimal generator matrix $\mathbf{\Psi}$. In this case, and in subsequent

examples, $M = 10000$. The 28×28 expanded degradation rate matrix is

$$\mathbf{\Lambda} = \begin{pmatrix} \mathbf{\Lambda}_{11} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{\Lambda}_{22} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{\Lambda}_{33} \end{pmatrix},$$

where

$$\mathbf{\Lambda}_{11} = [\lambda_{ij}] \text{ for } i = 1, \ldots, 5, j = 1, \ldots, 5,$$

where

$$\lambda_{ij} = \begin{cases} 0.1, & i = j \\ 0.0 & \text{otherwise} \end{cases},$$

$$\mathbf{\Lambda}_{22} = [\lambda_{ij}] \text{ for } i = 1, \ldots, 9, j = 1, \ldots, 9,$$

where

$$\lambda_{ij} = \begin{cases} 1.0, & i = j \\ 0.0 & \text{otherwise} \end{cases},$$

and

$$\mathbf{\Lambda}_{33} = [\lambda_{ij}] \text{ for } i = 1, \ldots, 14, j = 1, \ldots, 14,$$

where

$$\lambda_{ij} = \begin{cases} 2.0, & i = j \\ 0.0 & \text{otherwise} \end{cases}.$$

Substituting the matrices $\mathbf{\Psi}, \mathbf{\Lambda}$, and α into Equation (3.45) gives the failure time distribution in the transform space. Application of the one-dimensional Laplace transform inversion algorithm of Abate and Whitt [1] is used to arrive at the cumulative distribution values for selected points in time. The results of the failure time distribution are compared to results obtained from a simulation model in Table 4.2.

Table 4.2 Cumulative probability values for the brake pad example.

t	Simulated	Analytical	t	Simulated	Analytical
24.005	0.009900	0.009974	29.855	0.698370	0.698667
24.155	0.012130	0.012492	30.005	0.720320	0.720465
24.305	0.015210	0.015505	30.155	0.740820	0.741360
24.455	0.019080	0.019079	30.305	0.760130	0.761315
24.605	0.023290	0.023282	30.455	0.779130	0.780300
24.755	0.028480	0.028184	30.605	0.796970	0.798299
24.905	0.034180	0.033854	30.755	0.814180	0.815301
25.055	0.040840	0.040362	30.905	0.830210	0.831305
25.205	0.047980	0.047774	31.055	0.845830	0.846318
25.355	0.055930	0.056155	31.205	0.860470	0.860351
25.505	0.065210	0.065562	31.355	0.873860	0.873425
25.655	0.075500	0.076048	31.505	0.886670	0.885565
25.805	0.086910	0.087657	31.655	0.897460	0.896800
25.955	0.099610	0.100424	31.805	0.908100	0.907164
26.105	0.113240	0.114374	31.955	0.917590	0.916693
26.255	0.127900	0.129521	32.105	0.926270	0.925428
26.405	0.144520	0.145867	32.255	0.934120	0.933408
26.555	0.161680	0.163401	32.405	0.941310	0.940677
26.705	0.181090	0.182100	32.555	0.948080	0.947278
27.155	0.244520	0.244751	32.705	0.953890	0.953254
27.305	0.266340	0.267615	32.855	0.959580	0.958649
27.455	0.290150	0.291337	33.005	0.964200	0.963504
27.605	0.315800	0.315824	33.155	0.968540	0.967862
27.755	0.341430	0.340974	33.305	0.972330	0.971761
27.905	0.366960	0.366677	33.455	0.975700	0.975241
28.055	0.393380	0.392819	33.605	0.978610	0.978338
28.205	0.420010	0.419282	33.755	0.981160	0.981087
28.355	0.445410	0.445945	33.905	0.983770	0.983520
28.505	0.472630	0.472687	34.055	0.985840	0.985669
28.655	0.500000	0.499389	34.205	0.987830	0.987562
28.805	0.526700	0.525934	34.355	0.989590	0.989224
28.955	0.552120	0.552208	34.505	0.990880	0.990681
29.105	0.578320	0.578105	34.655	0.992160	0.991955
29.255	0.603280	0.603522	34.805	0.993230	0.993065
29.405	0.627850	0.628368	34.955	0.994270	0.994031
29.555	0.651890	0.652556	35.105	0.995220	0.994870
29.705	0.675710	0.676011	35.555	0.996890	0.996760

Based on this comparison, the technique developed in this thesis provides similar results to those computed via simulation. In fact, the maximum absolute deviation (MAD) in probability for this example is 0.001721. Figure 4.1 gives a graphical comparison of the cumulative distribution values at various time points.

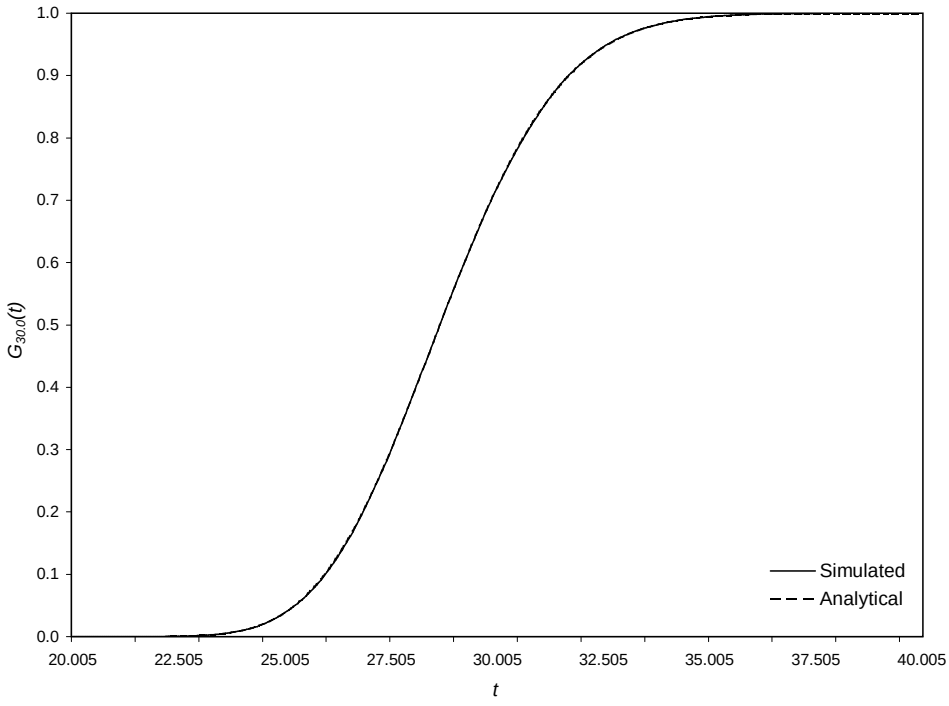

Figure 4.1 Simulated versus analytical cumulative distribution functions for the brake pad example.

Substituting the matrices $\mathbf{\Psi}, \mathbf{\Lambda}$, and α into Equation (3.46) gives the moments of failure time in the transform space. Application of the one-dimensional Laplace transform inversion algorithm of Abate and Whitt [1] is used to obtain the first and second moments of failure time, which are compared to results obtained via simulation in Table 4.3.

Table 4.3 Lower moments of failure time for the brake pad example.

Measure	Simulated	Analytical
$E[T_x]$	28.7615	28.7522
$E[T_x^2]$	832.2501	831.8517

4.2 Crack Propagation in a Turbine Blade

In this example, the propagation of a crack in a turbine blade of an aircraft engine is considered. Assume the turbine blade begins operation in perfect working order but develops a crack over time due to normal wear.

Let $X(t)$ denote the length of the crack at time t. It is assumed that the crack grows at a linear rate determined exclusively by the current state of the random environment to which the turbine blade is exposed. This random environment can be characterized as a four-state, semi-Markov process, $\{Z(t) : t \geq 0\}$, whose four states are defined by different ranges of turbine rotation speed. Moreover, the rates at which the different states cause the crack to propagate are known based on previous testing. The turbine blade fails when the crack reaches length $x = 20.0$. The state descriptions, state holding time distributions, and crack growth rates are listed in Table 4.4. Additionally, the transition probability matrix of the embedded DTMC is given as

$$\mathbf{P} = \begin{pmatrix} 0.0 & 0.6 & 0.2 & 0.2 \\ 0.5 & 0.0 & 0.4 & 0.1 \\ 0.2 & 0.4 & 0.0 & 0.4 \\ 0.1 & 0.3 & 0.6 & 0.0 \end{pmatrix}.$$

Table 4.4 Description of state space and crack growth rates for the turbine blade example.

State	Rotation Speed (rpm)	Holding Time Distribution	Crack Growth Rate
1	1000 through 4999	Beta(3,5)	0.17
2	5000 through 8999	Beta(2,4)	0.43
3	9000 through 12999	Weibull(2,5)	0.75
4	13000 through 15999	Weibull(3,6)	1.29

From Table 4.4, the matrix of degradation rates is

$$\mathbf{R}_D = \begin{pmatrix} 0.17 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.43 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.75 & 0.00 \\ 0.00 & 0.00 & 0.00 & 1.29 \end{pmatrix}.$$

It is assumed that the environment process begins in the first state; therefore, the initial probability vector is

$$\alpha = \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix}.$$

After observing the system over $\tau = 10000$ time units, the $\mathbf{T}_i$ matrices and $\mathbf{T}_i^o$ vectors of the phase-type representations of the four holding time distributions are calculated as

$$\mathbf{T}_1 = [\delta_{ij}] \text{ for } i = 1, \ldots, 6, j = 1, \ldots, 6,$$

where

$$\delta_{ij} = \begin{cases} -15.6780, & i = j \\ 15.3140, & i = 1,\ j = 2 \\ 15.6780, & i = j-1,\ j = 3, \ldots, 6 \\ 0 & \text{otherwise} \end{cases},$$

$$\mathbf{T}_1^o = [\gamma_i] \text{ for } i = 1, \ldots, 6,$$

where

$$\gamma_i = \begin{cases} 0.3638, & i = 1 \\ 0, & i = 2, \ldots, 5 \\ 15.6780, & i = 6 \end{cases},$$

$$\mathbf{T}_2 = [\delta_{ij}] \text{ for } i = 1, \ldots, 4, j = 1, \ldots, 4,$$

where

$$\delta_{ij} = \begin{cases} -11.6220, & i = j \\ 11.0970, & i = 1,\, j = 2 \\ 11.6220, & i = j-1,\, j = 3,4 \\ 0 & \text{otherwise} \end{cases},$$

$$\mathbf{T}_2^o = [\gamma_i] \text{ for } i = 1, \ldots, 4,$$

where

$$\gamma_i = \begin{cases} 0.5254, & i = 1 \\ 0, & i = 2,3 \\ 11.6220, & i = 4 \end{cases},$$

$$\mathbf{T}_3 = [\delta_{ij}] \text{ for } i = 1, \ldots, 4, j = 1, \ldots, 4,$$

where

$$\delta_{ij} = \begin{cases} -9.8405, & i = j \\ 9.5442, & i = 1,\, j = 2 \\ 9.8405, & i = j-1,\, j = 3,4 \\ 0 & \text{otherwise} \end{cases},$$

$$\mathbf{T}_3^o = [\gamma_i] \text{ for } i = 1, \ldots, 4,$$

where

$$\gamma_i = \begin{cases} 0.2964, & i = 1 \\ 0, & i = 2, \ldots, 3 \\ 9.8405, & i = 4 \end{cases},$$

$$\mathbf{T}_4 = [\delta_{ij}] \text{ for } i = 1, \ldots, 8, j = 1, \ldots, 8,$$

where

$$\delta_{ij} = \begin{cases} -16.1640, & i = j \\ 16.0280, & i = 1,\, j = 2 \\ 16.1640, & i = j-1,\, j = 3, \ldots, 8 \\ 0 & \text{otherwise} \end{cases},$$

and

$$\mathbf{T}_4^o = [\gamma_i] \text{ for } i = 1, \ldots, 8,$$

where

$$\gamma_i = \begin{cases} 0.1357, & i = 1 \\ 0, & i = 2, \ldots, 7 \\ 16.1640, & i = 8 \end{cases}.$$

Furthermore, the observed transition rates among the four environment states are presented as the elements of $\hat{\mathbf{Q}}$, where

$$\hat{\mathbf{Q}} = \begin{pmatrix} -2.6646 & 1.5965 & 0.5313 & 0.5368 \\ 1.5020 & -3.0075 & 1.1991 & 0.3064 \\ 0.5060 & 1.0083 & -2.5170 & 1.0027 \\ 0.2033 & 0.6087 & 1.2234 & -2.0354 \end{pmatrix}.$$

Using the above phase-type representations and the $\hat{q}_{ij}$ elements of $\hat{\mathbf{Q}}$, the 26×26 generator matrix of the newly formed CTMC is

$$\mathbf{\Psi} = \begin{pmatrix} \mathbf{\Psi}_{11} & \mathbf{\Psi}_{12} & \mathbf{\Psi}_{13} & \mathbf{\Psi}_{14} \\ \mathbf{\Psi}_{21} & \mathbf{\Psi}_{22} & \mathbf{\Psi}_{23} & \mathbf{\Psi}_{24} \\ \mathbf{\Psi}_{31} & \mathbf{\Psi}_{32} & \mathbf{\Psi}_{33} & \mathbf{\Psi}_{34} \\ \mathbf{\Psi}_{41} & \mathbf{\Psi}_{42} & \mathbf{\Psi}_{43} & \mathbf{\Psi}_{44} \end{pmatrix},$$

where

$$\mathbf{\Psi}_{11} = \begin{pmatrix} \mathbf{T}_1 & \mathbf{T}_1^o \\ \mathbf{0} & -26646.0 \end{pmatrix},$$

$$\mathbf{\Psi}_{12} = [\psi_{ij}] \text{ for } i = 1, \ldots, 7, j = 1, \ldots, 5,$$

where

$$\psi_{ij} = \begin{cases} 15965.0, & i = 7,\ j = 1 \\ 0 & \text{otherwise} \end{cases},$$

$$\mathbf{\Psi}_{13} = [\psi_{ij}] \text{ for } i = 1, \ldots, 7, j = 1, \ldots, 5,$$

where

$$\psi_{ij} = \begin{cases} 5312.6, & i = 7,\ j = 1 \\ 0 & \text{otherwise} \end{cases}, \textit{and}$$

$$\mathbf{\Psi}_{14} = [\psi_{ij}] \text{ for } i = 1, \ldots, 7, j = 1, \ldots, 9,$$

where

$$\psi_{ij} = \begin{cases} 5368.4, & i = 7,\ j = 1 \\ 0 & \text{otherwise} \end{cases},$$

$$\mathbf{\Psi}_{21} = [\psi_{ij}] \text{ for } i = 1, \ldots, 5, j = 1, \ldots, 7,$$

where

$$\psi_{ij} = \begin{cases} 15020.0, & i = 5,\ j = 1 \\ 0 & \text{otherwise} \end{cases},$$

$$\mathbf{\Psi}_{22} = \begin{pmatrix} \mathbf{T}_2 & \mathbf{T}_2^o \\ \mathbf{0} & -30075.0 \end{pmatrix},$$

$$\mathbf{\Psi}_{23} = [\psi_{ij}] \text{ for } i = 1, \ldots, 5, j = 1, \ldots, 5,$$

where

$$\psi_{ij} = \begin{cases} 11991.0, & i = 5,\ j = 1 \\ 0 & \text{otherwise} \end{cases},$$

$$\mathbf{\Psi}_{24} = [\psi_{ij}] \text{ for } i = 1, \ldots, 5, j = 1, \ldots, 9,$$

where

$$\psi_{ij} = \begin{cases} 3063.6, & i = 5,\ j = 1 \\ 0 & \text{otherwise} \end{cases},$$

$$\mathbf{\Psi}_{31} = [\psi_{ij}] \text{ for } i = 1, \ldots, 5, j = 1, \ldots, 7,$$

where

$$\psi_{ij} = \begin{cases} 5059.6, & i = 5,\ j = 1 \\ 0 & \text{otherwise} \end{cases},$$

$$\mathbf{\Psi}_{32} = [\psi_{ij}] \ \text{ for } \ i = 1, \ldots, 5, j = 1, \ldots, 5,$$

where

$$\psi_{ij} = \begin{cases} 10083.0, & i = 5,\ j = 1 \\ 0 & \text{otherwise} \end{cases},$$

$$\mathbf{\Psi}_{33} = \begin{pmatrix} \mathbf{T}_3 & \mathbf{T}_3^o \\ \mathbf{0} & -25170.0 \end{pmatrix},$$

and

$$\mathbf{\Psi}_{34} = [\psi_{ij}] \ \text{ for } \ i = 1, \ldots, 5, j = 1, \ldots, 9,$$

where

$$\psi_{ij} = \begin{cases} 10027.0, & i = 5,\ j = 1 \\ 0 & \text{otherwise} \end{cases},$$

$$\mathbf{\Psi}_{41} = [\psi_{ij}] \ \text{ for } \ i = 1, \ldots, 9, j = 1, \ldots, 7,$$

where

$$\psi_{ij} = \begin{cases} 2033.2, & i = 9,\ j = 1 \\ 0 & \text{otherwise} \end{cases},$$

$$\mathbf{\Psi}_{42} = [\psi_{ij}] \ \text{ for } \ i = 1, \ldots, 9, j = 1, \ldots, 5,$$

where

$$\psi_{ij} = \begin{cases} 6086.7, & i = 9,\ j = 1 \\ 0 & \text{otherwise} \end{cases},$$

$$\mathbf{\Psi}_{43} = [\psi_{ij}] \ \text{ for } \ i = 1, \ldots, 9, j = 1, \ldots, 5,$$

where

$$\psi_{ij} = \begin{cases} 12234.0, & i = 9,\ j = 1 \\ 0 & \text{otherwise} \end{cases},$$

and

$$\mathbf{\Psi}_{44} = \begin{pmatrix} \mathbf{T}_4 & \mathbf{T}_4^o \\ \mathbf{0} & -20354.0 \end{pmatrix}.$$

The 26×26 expanded degradation rate matrix is

$$\mathbf{\Lambda} = \begin{pmatrix} \mathbf{\Lambda}_{11} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{\Lambda}_{22} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{\Lambda}_{33} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{\Lambda}_{44} \end{pmatrix},$$

where

$$\mathbf{\Lambda}_{11} = [\lambda_{ij}] \text{ for } i = 1, \ldots, 7, j = 1, \ldots, 7,$$

where

$$\lambda_{ij} = \begin{cases} 1.7, & i = j \\ 0 & \text{otherwise} \end{cases},$$

$$\mathbf{\Lambda}_{22} = [\lambda_{ij}] \text{ for } i = 1, \ldots, 5, j = 1, \ldots, 5,$$

where

$$\lambda_{ij} = \begin{cases} 4.3, & i = j \\ 0 & \text{otherwise} \end{cases},$$

$$\mathbf{\Lambda}_{33} = [\lambda_{ij}] \text{ for } i = 1, \ldots, 5, j = 1, \ldots, 5,$$

where

$$\lambda_{ij} = \begin{cases} 7.5, & i = j \\ 0 & \text{otherwise} \end{cases},$$

and

$$\mathbf{\Lambda}_{44} = [\lambda_{ij}] \text{ for } i = 1, \ldots, 9, j = 1, \ldots, 9,$$

where

$$\lambda_{ij} = \begin{cases} 12.9, & i = j \\ 0 & \text{otherwise} \end{cases}.$$

Substituting the matrices $\mathbf{\Psi}, \mathbf{\Lambda}$, and α into Equation (3.45) gives the solution of the failure time distribution in the transform space. Application of the one-dimensional Laplace transform inversion algorithm of Abate and Whitt [1] is again used to arrive at the cumulative distribution values for selected points in time. The results of the failure time distribution are compared to those obtained from a simulation model in Table 4.5. Based on this comparison, the technique developed in this thesis provides similar results to those computed via simulation. In fact, the MAD in probability for this example is 0.003297. Figure 4.2 gives a graphical comparison of the cumulative distribution values at various time points. Substituting the matrices $\mathbf{\Psi}, \mathbf{\Lambda}$, and

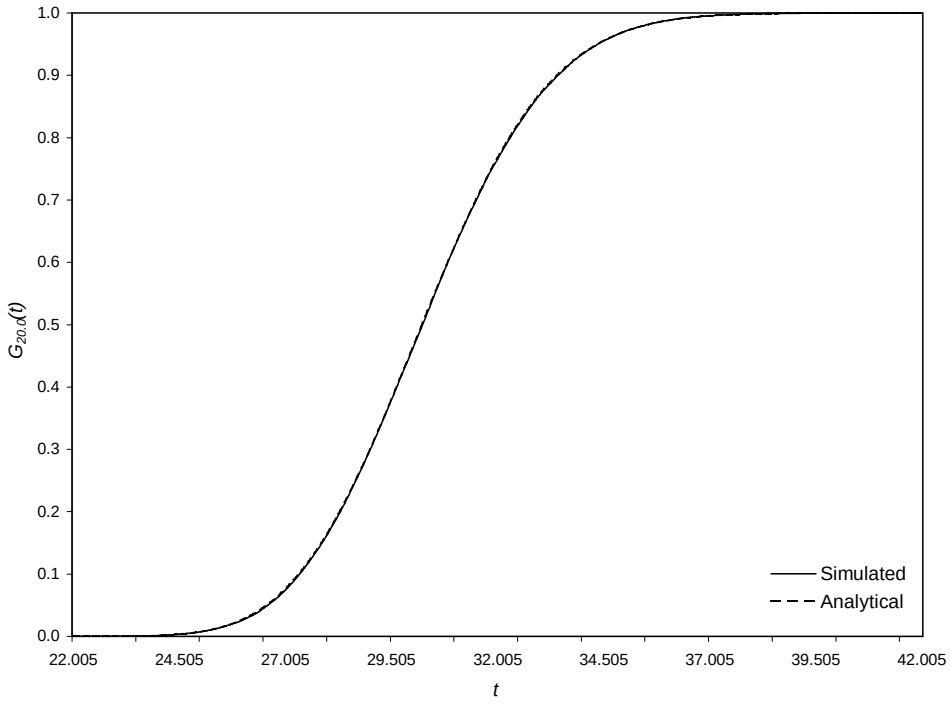

Figure 4.2 Simulated versus analytical cumulative distribution functions for the turbine blade example.

α into Equation (3.46) gives the moments of failure time in the transform space. Application of the one-dimensional Laplace transform inversion algorithm of Abate

Table 4.5 Cumulative probability values for the turbine blade example.

t	Simulated	Analytical	t	Simulated	Analytical
25.605	0.016640	0.016979	32.005	0.765210	0.767488
25.805	0.021170	0.021731	32.205	0.788620	0.791122
26.005	0.026420	0.027490	32.405	0.811180	0.813169
26.205	0.032860	0.034391	32.605	0.831400	0.833613
26.405	0.040910	0.042564	32.805	0.850860	0.852463
26.605	0.050410	0.052139	33.005	0.869020	0.869743
26.805	0.061490	0.063234	33.205	0.884060	0.885496
27.005	0.074020	0.075958	33.405	0.898080	0.899777
27.205	0.088680	0.090403	33.605	0.910950	0.912653
27.405	0.104930	0.106641	33.805	0.923320	0.924198
27.605	0.123250	0.124720	34.005	0.933560	0.934495
27.805	0.143160	0.144660	34.205	0.942700	0.943631
28.005	0.164710	0.166455	34.405	0.951060	0.951694
28.205	0.187980	0.190066	34.605	0.958330	0.958773
28.405	0.213170	0.215423	34.805	0.964630	0.964957
28.605	0.240960	0.242425	35.005	0.970190	0.970332
28.805	0.269260	0.270942	35.205	0.974880	0.974981
29.005	0.299300	0.300818	35.405	0.978940	0.978982
29.205	0.330490	0.331868	35.605	0.982560	0.982410
29.405	0.362360	0.363891	35.805	0.985580	0.985332
29.605	0.395080	0.396667	36.005	0.987980	0.987811
29.805	0.428160	0.429965	36.205	0.990040	0.989906
30.005	0.461350	0.463548	36.405	0.992030	0.991668
30.205	0.493880	0.497177	36.605	0.993350	0.993143
30.405	0.527590	0.530617	36.805	0.994600	0.994374
30.605	0.561840	0.563640	37.005	0.995600	0.995395
30.805	0.594630	0.596032	37.205	0.996300	0.996239
31.005	0.626510	0.627594	37.405	0.996860	0.996934
31.205	0.656950	0.658147	37.605	0.997450	0.997503
31.405	0.685580	0.687534	37.805	0.997990	0.997966
31.605	0.713420	0.715622	38.005	0.998490	0.998342
31.805	0.740940	0.742302	38.205	0.998750	0.998644

and Whitt [1] is used to obtain the first and second moments of failure time, which are compared to results obtained via simulation in Table 4.6.

Table 4.6 Lower moments of failure time for the turbine blade example.

Measure	Simulated	Analytical
$E[T_x]$	30.3253	30.3046
$E[T_x^2]$	925.2438	924.0687

4.3 Chemical Decomposition of an Automotive Coating

In this final example, consider an automotive coating that is subject to weathering by the outdoor environment. Environmental effects such as temperature and solar radiation cause chemical decomposition of the coating, which can be measured in terms of gloss loss and/or color change [7]. When the cumulative decomposition of the coating reaches a certain threshold x, the coating is said to have reached failure.

Assume the chemical decomposition (degradation) rate is linear and depends strictly on the random state of the outdoor weather environment to which the coating is exposed. When the cumulative chemical decomposition of the coating reaches the value $x = 5.0$, the coating is said to have failed. The weather environment, whose next state depends only upon the current state, can be characterized as a semi-Markov process and consists of five distinct states. These states, their definitions, the probability distributions of their holding times, and their associated decomposition rates are displayed in Table 4.7. Additionally, the transition probability matrix of

Table 4.7 Description of state space and decomposition rates for the coating example.

State	Sky	Temp	Holding Times	Decomposition Rate
1	Cloudy	$\leq$ 32 deg F	Weibull(3,5)	0.46
2	Cloudy	$>$ 32 deg F	Beta(2,5)	0.82
3	Sunny	$\leq$ 32 deg F	Weibull(4,6)	1.10
4	Sunny	$>$ 32 deg F	Beta(6,3)	1.34
5	Rain	$>$ 32 deg F	Gamma(.5,.1)	1.98

the embedded DTMC is given as

$$\mathbf{P} = \begin{pmatrix} 0.0 & 0.5 & 0.2 & 0.2 & 0.1 \\ 0.5 & 0.0 & 0.3 & 0.1 & 0.1 \\ 0.2 & 0.4 & 0.0 & 0.2 & 0.2 \\ 0.1 & 0.1 & 0.2 & 0.0 & 0.6 \\ 0.2 & 0.1 & 0.1 & 0.6 & 0.0 \end{pmatrix}.$$

From Table 4.7, the matrix of degradation rates is

$$\mathbf{R}_D = \begin{pmatrix} 0.46 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.82 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 1.10 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 1.34 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 1.98 \end{pmatrix}.$$

Again, it is assumed that the environment process begins in the first state; therefore, the initial probability vector is

$$\alpha = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

After observing the system over $\tau = 40000$ time units, the $\mathbf{T}_i$ matrices and $\mathbf{T}_i^o$ vectors of the phase-type representations of the five holding time distributions are calculated as

$$\mathbf{T}_1 = [\delta_{ij}] \text{ for } i = 1, \ldots, 6, j = 1, \ldots, 6,$$

where

$$\delta_{ij} = \begin{cases} -15.694, & i = j \\ 15.328, & i = 1,\ j = 2 \\ 15.694, & i = j - 1,\ j = 3, \ldots, 6 \\ 0 & \text{otherwise} \end{cases},$$

$$\mathbf{T}_1^o = [\gamma_i] \text{ for } i = 1, \ldots, 6,$$

where

$$\gamma_i = \begin{cases} 0.36645, & i = 1 \\ 0, & i = 2, \ldots, 5 \\ 15.694, & i = 6 \end{cases},$$

$$\mathbf{T}_2 = [\delta_{ij}] \text{ for } i = 1, \ldots, 4, j = 1, \ldots, 4,$$

where

$$\delta_{ij} = \begin{cases} -9.8485, & i = j \\ 9.602, & i = 1,\ j = 2 \\ 9.8485, & i = j - 1,\ j = 3, 4 \\ 0 & \text{otherwise} \end{cases},$$

$$\mathbf{T}_2^o = [\gamma_i] \text{ for } i = 1, \ldots, 4,$$

where

$$\gamma_i = \begin{cases} 0.24652, & i = 1 \\ 0, & i = 2, \ldots, 3 \\ 9.8485, & i = 4 \end{cases},$$

$$\mathbf{T}_3 = [\delta_{ij}] \text{ for } i = 1, \ldots, 8, j = 1, \ldots, 8,$$

where

$$\delta_{ij} = \begin{cases} -19.704, & i = j \\ 19.444, & i = 1,\ j = 2 \\ 19.704, & i = j - 1,\ j = 3, \ldots, 8 \\ 0 & \text{otherwise} \end{cases},$$

$$\mathbf{T}_3^o = [\gamma_i] \text{ for } i = 1, \ldots, 8,$$

where

$$\gamma_i = \begin{cases} 0.25978, & i = 1 \\ 0, & i = 2, \ldots, 7 \\ 19.704, & i = 8 \end{cases},$$

$$\mathbf{T}_4 = [\delta_{ij}] \text{ for } i = 1, \ldots, 8, j = 1, \ldots, 8,$$

where

$$\delta_{ij} = \begin{cases} -16.122, & i = j \\ 16.01, & i = 1,\ j = 2 \\ 16.122, & i = j - 1,\ j = 3, \ldots, 8 \\ 0 & \text{otherwise} \end{cases},$$

$$\mathbf{T}_4^o = [\gamma_i] \text{ for } i = 1, \ldots, 8,$$

where

$$\gamma_i = \begin{cases} 0.11148, & i = 1 \\ 0, & i = 2, \ldots, 7 \\ 16.122, & i = 8 \end{cases},$$

$$\mathbf{T}_5 = [\delta_{ij}] \text{ for } i = 1, \ldots, 2, j = 1, \ldots, 2,$$

where

$$\delta_{ij} = \begin{cases} -22.404, & i = j \\ 0.43763, & i = 1,\ j = 2 \\ -3.6311, & i = 2,\ j = 2 \\ 0 & \text{otherwise} \end{cases},$$

and

$$\mathbf{T}_5^o = [\gamma_i] \text{ for } i = 1, 2,$$

where

$$\gamma_i = \begin{cases} 21.967, & i = 1 \\ 3.6311, & i = 2 \end{cases}.$$

Furthermore, the observed transition rates among the five environment states are presented as the elements of $\hat{\mathbf{Q}}$, where

$$\hat{\mathbf{Q}} = \begin{pmatrix} -2.6675 & 1.3358 & 0.5379 & 0.5336 & 0.2603 \\ 1.2608 & -2.5092 & 0.7500 & 0.2502 & 0.2483 \\ 0.5079 & 0.9927 & -2.4917 & 0.5004 & 0.4907 \\ 0.2001 & 0.2033 & 0.4167 & -2.0275 & 1.2074 \\ 4.0752 & 1.9409 & 1.9833 & 11.9950 & -19.9940 \end{pmatrix}.$$

Using the above phase-type representations, the $\hat{q}_{ij}$ elements of $\hat{\mathbf{Q}}$, and $M = 10000$, the 33×33 generator matrix of the newly formed CTMC is

$$\mathbf{\Psi} = \begin{pmatrix} \mathbf{\Psi}_{11} & \mathbf{\Psi}_{12} & \mathbf{\Psi}_{13} & \mathbf{\Psi}_{14} & \mathbf{\Psi}_{15} \\ \mathbf{\Psi}_{21} & \mathbf{\Psi}_{22} & \mathbf{\Psi}_{23} & \mathbf{\Psi}_{24} & \mathbf{\Psi}_{25} \\ \mathbf{\Psi}_{31} & \mathbf{\Psi}_{32} & \mathbf{\Psi}_{33} & \mathbf{\Psi}_{34} & \mathbf{\Psi}_{35} \\ \mathbf{\Psi}_{41} & \mathbf{\Psi}_{42} & \mathbf{\Psi}_{43} & \mathbf{\Psi}_{44} & \mathbf{\Psi}_{45} \\ \mathbf{\Psi}_{51} & \mathbf{\Psi}_{52} & \mathbf{\Psi}_{53} & \mathbf{\Psi}_{54} & \mathbf{\Psi}_{55} \end{pmatrix},$$

where

$$\mathbf{\Psi}_{11} = \begin{pmatrix} \mathbf{T}_1 & \mathbf{T}_1^o \\ \mathbf{0} & -26675.0 \end{pmatrix},$$

$$\mathbf{\Psi}_{12} = [\psi_{ij}] \text{ for } i = 1, \ldots, 7, j = 1, \ldots, 5,$$

where

$$\psi_{ij} = \begin{cases} 13358.0, & i = 7,\ j = 1 \\ 0 & \text{otherwise} \end{cases},$$

$$\mathbf{\Psi}_{13} = [\psi_{ij}] \text{ for } i = 1, \ldots, 7, j = 1, \ldots, 9,$$

where

$$\psi_{ij} = \begin{cases} 5378.7, & i = 7,\ j = 1 \\ 0 & \text{otherwise} \end{cases}, \textit{and}$$

$$\mathbf{\Psi}_{14} = [\psi_{ij}] \text{ for } i = 1, \ldots, 7, j = 1, \ldots, 9,$$

where

$$\psi_{ij} = \begin{cases} 5335.6, & i = 7, j = 1 \\ 0 & \text{otherwise} \end{cases},$$

$$\mathbf{\Psi}_{15} = [\psi_{ij}] \text{ for } i = 1, \ldots, 7, j = 1, \ldots, 3,$$

where

$$\psi_{ij} = \begin{cases} 2602.6, & i = 7,\ j = 1 \\ 0 & \text{otherwise} \end{cases},$$

$$\mathbf{\Psi}_{21} = [\psi_{ij}] \text{ for } i = 1, \ldots, 5, j = 1, \ldots, 7,$$

where

$$\psi_{ij} = \begin{cases} 12608.0, & i = 5,\ j = 1 \\ 0 & \text{otherwise} \end{cases},$$

$$\mathbf{\Psi}_{22} = \begin{pmatrix} \mathbf{T}_2 & \mathbf{T}_2^o \\ \mathbf{0} & -25092.0 \end{pmatrix},$$

$$\mathbf{\Psi}_{23} = [\psi_{ij}] \text{ for } i = 1, \ldots, 5, j = 1, \ldots, 9,$$

where

$$\psi_{ij} = \begin{cases} 7499.9, & i = 5,\ j = 1 \\ 0 & \text{otherwise} \end{cases},$$

$$\mathbf{\Psi}_{24} = [\psi_{ij}] \text{ for } i = 1, \ldots, 5, j = 1, \ldots, 9,$$

where

$$\psi_{ij} = \begin{cases} 2502.0, & i = 5,\ j = 1 \\ 0 & \text{otherwise} \end{cases},$$

$$\mathbf{\Psi}_{25} = [\psi_{ij}] \text{ for } i = 1, \ldots, 5, j = 1, \ldots, 3,$$

where

$$\psi_{ij} = \begin{cases} 2482.5, & i = 5,\ j = 1 \\ 0 & \text{otherwise} \end{cases},$$

$$\mathbf{\Psi}_{31} = [\psi_{ij}] \text{ for } i = 1, \ldots, 9, j = 1, \ldots, 7,$$

where

$$\psi_{ij} = \begin{cases} 5079.3, & i = 9,\ j = 1 \\ 0 & \text{otherwise} \end{cases},$$

$$\mathbf{\Psi}_{32} = [\psi_{ij}] \text{ for } i = 1, \ldots, 9, j = 1, \ldots, 5,$$

where

$$\psi_{ij} = \begin{cases} 9926.7, & i = 9,\ j = 1 \\ 0 & \text{otherwise} \end{cases},$$

$$\mathbf{\Psi}_{33} = \begin{pmatrix} \mathbf{T}_3 & \mathbf{T}_3^o \\ \mathbf{0} & -24917.0 \end{pmatrix},$$

and

$$\mathbf{\Psi}_{34} = [\psi_{ij}] \text{ for } i = 1, \ldots, 9, j = 1, \ldots, 9,$$

where

$$\psi_{ij} = \begin{cases} 5004.1, & i = 9,\ j = 1 \\ 0 & \text{otherwise} \end{cases},$$

$$\mathbf{\Psi}_{35} = [\psi_{ij}] \text{ for } i = 1, \ldots, 9, j = 1, \ldots, 3,$$

where

$$\psi_{ij} = \begin{cases} 4907.3, & i = 9,\ j = 1 \\ 0 & \text{otherwise} \end{cases},$$

$$\mathbf{\Psi}_{41} = [\psi_{ij}] \text{ for } i = 1, \ldots, 9, j = 1, \ldots, 7,$$

where

$$\psi_{ij} = \begin{cases} 2000.5, & i = 9,\ j = 1 \\ 0 & \text{otherwise} \end{cases},$$

$$\mathbf{\Psi}_{42} = [\psi_{ij}] \text{ for } i = 1, \ldots, 9, j = 1, \ldots, 5,$$

where

$$\psi_{ij} = \begin{cases} 2033.4, & i = 9,\ j = 1 \\ 0 & \text{otherwise} \end{cases},$$

$$\mathbf{\Psi}_{43} = [\psi_{ij}] \ \text{ for } \ i = 1, \ldots, 9, j = 1, \ldots, 9,$$

where

$$\psi_{ij} = \begin{cases} 4167.4, & i = 9,\ j = 1 \\ 0 & \text{otherwise} \end{cases},$$

and

$$\mathbf{\Psi}_{44} = \begin{pmatrix} \mathbf{T}_4 & \mathbf{T}_4^o \\ \mathbf{0} & -20275.0 \end{pmatrix}.$$

$$\mathbf{\Psi}_{45} = [\psi_{ij}] \ \text{ for } \ i = 1, \ldots, 9, j = 1, \ldots, 3,$$

where

$$\psi_{ij} = \begin{cases} 12074.0, & i = 9,\ j = 1 \\ 0 & \text{otherwise} \end{cases},$$

$$\mathbf{\Psi}_{51} = [\psi_{ij}] \ \text{ for } \ i = 1, \ldots, 3, j = 1, \ldots, 7,$$

where

$$\psi_{ij} = \begin{cases} 40752.0, & i = 3,\ j = 1 \\ 0 & \text{otherwise} \end{cases},$$

$$\mathbf{\Psi}_{52} = [\psi_{ij}] \ \text{ for } \ i = 1, \ldots, 3, j = 1, \ldots, 5,$$

where

$$\psi_{ij} = \begin{cases} 19409.0, & i = 3,\ j = 1 \\ 0 & \text{otherwise} \end{cases},$$

$$\mathbf{\Psi}_{53} = [\psi_{ij}] \ \text{ for } \ i = 1, \ldots, 3, j = 1, \ldots, 9,$$

where

$$\psi_{ij} = \begin{cases} 19833.0, & i = 3,\ j = 1 \\ 0 & \text{otherwise} \end{cases}, \textit{and}$$

$$\mathbf{\Psi}_{54} = [\psi_{ij}] \ \text{ for } \ i = 1, \ldots, 3, j = 1, \ldots, 9,$$

where

$$\psi_{ij} = \begin{cases} 119950.0000, & i = 3,\, j = 1 \\ 0 & \text{otherwise} \end{cases},$$

and

$$\mathbf{\Psi}_{55} = \begin{pmatrix} \mathbf{T}_5 & \mathbf{T}_5^o \\ \mathbf{0} & -199940.0 \end{pmatrix}.$$

The 33×33 expanded degradation rate matrix is

$$\mathbf{\Lambda} = \begin{pmatrix} \mathbf{\Lambda}_{11} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{\Lambda}_{22} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{\Lambda}_{33} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{\Lambda}_{44} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{\Lambda}_{55} \end{pmatrix},$$

where

$$\mathbf{\Lambda}_{11} = [\lambda_{ij}] \text{ for } i = 1, \ldots, 7, j = 1, \ldots, 7,$$

where

$$\lambda_{ij} = \begin{cases} 2.3, & i = j \\ 0 & \text{otherwise} \end{cases},$$

$$\mathbf{\Lambda}_{22} = [\lambda_{ij}] \text{ for } i = 1, \ldots, 5, j = 1, \ldots, 5,$$

where

$$\lambda_{ij} = \begin{cases} 4.1, & i = j \\ 0 & \text{otherwise} \end{cases},$$

$$\mathbf{\Lambda}_{33} = [\lambda_{ij}] \text{ for } i = 1, \ldots, 9, j = 1, \ldots, 9,$$

where

$$\lambda_{ij} = \begin{cases} 5.5, & i = j \\ 0 & \text{otherwise} \end{cases},$$

and

$$\mathbf{\Lambda}_{44} = [\lambda_{ij}] \text{ for } i = 1, \ldots, 9, j = 1, \ldots, 9,$$

where

$$\lambda_{ij} = \begin{cases} 6.7, & i = j \\ 0 & \text{otherwise} \end{cases},$$

and

$$\mathbf{\Lambda}_{55} = [\lambda_{ij}] \text{ for } i = 1, \ldots, 3, j = 1, \ldots, 3,$$

where

$$\lambda_{ij} = \begin{cases} 9.9, & i = j \\ 0 & \text{otherwise} \end{cases}.$$

Applying $\mathbf{\Psi}, \mathbf{\Lambda}$, and α to Equation (3.45) gives the solution of the failure time distribution in the transform space. The analytical results of the failure time distribution in the time domain are compared to results derived via simulation in Table 4.8. Based on this comparison, the technique developed in this thesis provides similar results to those computed via simulation. In fact, the MAD in probability for this example is 0.004181. Figure 4.3 gives a graphical comparison of the cumulative distribution values at various time points.

Table 4.8 Cumulative probability values for the coating example.

t	Simulated	Analytical	t	Simulated	Analytical
4.155	0.021700	0.022164	5.640	0.726490	0.723197
4.210	0.029530	0.029350	5.695	0.754040	0.750956
4.265	0.038490	0.038055	5.750	0.780140	0.777106
4.320	0.049100	0.048431	5.805	0.804230	0.801577
4.375	0.061180	0.060610	5.860	0.827300	0.824326
4.430	0.075030	0.074701	5.915	0.847530	0.845333
4.485	0.091160	0.090782	5.970	0.866250	0.864603
4.540	0.109060	0.108900	6.025	0.883390	0.882159
4.595	0.128770	0.129070	6.080	0.898950	0.898048
4.650	0.150780	0.151274	6.135	0.912910	0.912330
4.705	0.174850	0.175464	6.190	0.925670	0.925079
4.760	0.201460	0.201560	6.245	0.937570	0.936383
4.815	0.228750	0.229453	6.300	0.947580	0.946335
4.870	0.259000	0.259001	6.355	0.955760	0.955035
4.925	0.291560	0.290033	6.410	0.963040	0.962588
4.980	0.323730	0.322352	6.465	0.968900	0.969095
5.035	0.356320	0.355736	6.520	0.974440	0.974662
5.090	0.390980	0.389948	6.575	0.978930	0.979388
5.145	0.425950	0.424735	6.630	0.983290	0.983370
5.200	0.461860	0.459840	6.685	0.986990	0.986698
5.255	0.496930	0.495004	6.740	0.989770	0.989457
5.310	0.532100	0.529974	6.795	0.991880	0.991726
5.365	0.567740	0.564506	6.850	0.993720	0.993576
5.420	0.601350	0.598370	6.905	0.995140	0.995071
5.475	0.634620	0.631353	6.960	0.996270	0.996268
5.530	0.667410	0.663261	7.015	0.997180	0.997218
5.585	0.697890	0.693925	7.070	0.997820	0.997964

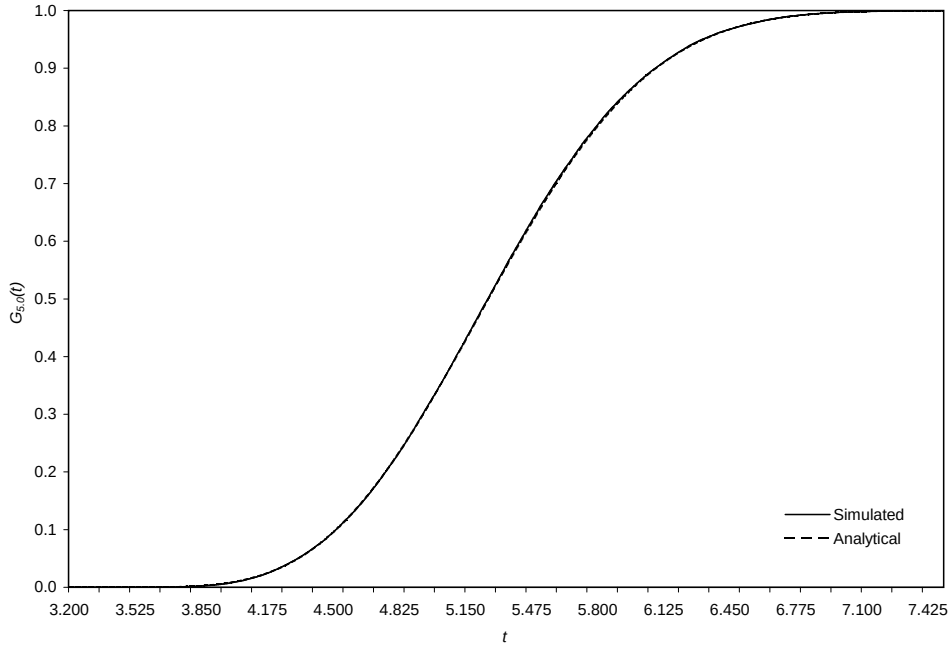

Figure 4.3 Simulated versus analytical cumulative distribution functions for the coating example.

Substituting the matrices $\mathbf{\Psi}, \mathbf{\Lambda}$, and α into Equation (3.46) gives the moments of failure time in the transform space. Application of the one-dimensional Laplace transform inversion algorithm of Abate and Whitt [1] is used to obtain the first and second moments of failure time, which are compared to results obtained via simulation in Table 4.9.

Table 4.9 Lower moments of failure time for the coating example.

Measure	Simulated	Analytical
$E[T_x]$	5.2830	5.2867
$E[T_x^2]$	28.2738	28.3145

The examples in this chapter show that the failure time distribution of a system experiencing wear due to an environment characterized as a semi-Markov process can be approximated by simply observing the environment over time. In all three numerical examples, the maximum absolute deviation (MAD) in probability was less

than or equal to 0.0042. Considering the inherent inaccuracies due to the use of phase-type approximations, the results are favorable. Assuming that the wear rates imposed on the system by each state are known, lifetime estimation can be done without monitoring the cumulative degradation level during the system's operating lifetime. This environment-based method is particularly advantageous over traditional degradation-based reliability techniques when the operating environment of the system is known or observable but degradation measurements are difficult or impossible to obtain.

5. Conclusions and Future Research

This thesis presented a reliability analysis technique for a system that experiences wear under the influence of a dynamic environment process. In particular, this effort provided a means to compute the failure time distribution and moments of failure time for a single-unit system whose wear rate depends on the state of its stochastic environment when the environment is characterized as a semi-Markov process. The numerical examples showed that a lifetime prediction can be made for a degrading system simply by observing the system's operating environment over time.

The first step in this research effort was to review the two main approaches to system lifetime estimation: classical failure time analysis and degradation-based reliability analysis. After a more through study of the current literature on degradation-based reliability techniques, it was observed that a common drawback of these approaches is the lack of consideration of the effects of the operating environment on the performance of the system. Since many systems operate under complex and changing conditions, a study of the effects of the environment on a system is necessary to obtain a realistic estimation of the system's reliability. One environment-based model produced good results, as compared to those from a simulation model, when the state-dependent wear rates were known and the environment was characterized as a finite state continuous-time Markov chain (CTMC) [16]. However, this model assumed that the time spent in each of the environment states is an exponentially distributed random variable. Under realistic conditions, the state holding time distributions may not be exponentially distributed or even known.

If the evolution of the environment is assumed to depend only upon the current state, then any such environment process has an embedded discrete-time Markov chain (DTMC) and can be characterized as a semi-Markov process. In such cases, knowledge of the holding times in each state may be gained solely by observation.

This thesis presented a technique in which an environment characterized as a semi-Markov process can be re-characterized as a continuous-time Markov chain. This was accomplished by approximating the state holding time distributions with phase-type distributions. An overview of phase-type approximation techniques was presented in chapter 3. The transition rates among the various states of the environment and the phase-type representations were then used to construct the infinitesimal generator matrix of the newly created CTMC. This generator matrix, the appropriately resized matrix of wear rates, and the initial probability vector of the environment process were used to compute the failure time distribution and moments of failure time of the system in the transform using the one-dimensional technique found in [17].

After the entire procedure was outlined in chapter 3, numerical examples were illustrated in chapter 4 to show the potential accuracy of failure time predictions using the phase-type approximation technique. In each example, the environment was observed for a sufficiently long time period during which the number of transitions among the various states and the state holding times were observed. From the transition observations, transition rates were estimated for each state of the environment. Phase-type representations of the state holding time distributions, computed using an algorithm implemented in MATLAB®, were then used with the transition rates to construct the infinitesimal generator matrix of the newly formed CTMC. The degradation rate matrix was expanded to account for the number of phases used to approximate each state. The expanded degradation rate matrix, the generator matrix, and the initial probability vector of the environment process were then used as inputs to Equations (3.45) and (3.46) to obtain the failure time distribution values and moments of failure time, respectively, in the transform space. A numerical inversion technique [1], implemented in MATLAB®, was then used to obtain distribution values and moments in the time domain. These values were compared to those obtained via simulation to show the relative accuracy of lifetime estimation. In all three examples, the maximum absolute deviation (MAD) in probability was less

than or equal to 0.0042. Considering that the infinitesimal generator matrix used in Equation (3.45) was based upon phase-type approximations of the state holding time distributions, these comparisons provide very favorable results.

The main contribution of this thesis effort is the resulting procedure for making lifetime predictions of degrading systems based solely upon observations of their operating environments. In particular, this work generalizes the results in [16] and [17] to the case where state holding time distributions are nonexponential or even unknown. By using phase-type distributions to approximate state holding time distributions, the method developed in this thesis eliminates the requirement for exponentially distributed holding times, allowing any environment that can be characterized as a semi-Markov process to be converted into a CTMC. Furthermore, this thesis expands the knowledge base of degradation-based reliability analysis by providing a technique for failure time estimation that does not rely upon degradation measurements during the lifetime of the system. This technique can prove particularly useful in situations where the system's operating environment is observable or known, but where observations of the cumulative wear to the system over time are not easy (or even possible) to obtain. For example, designers of components on satellites or systems that operate under deep water may benefit from this technique. Such systems may operate in predictable or observable environments, but restrictions due to weight, cost, or environmental conditions may prevent the collection of degradation data.

Suggested areas for future research include expansion of the phase-type approximation technique to those other than the Coxian and Erlang distributions. These distributions were used due to their simplicity and the fact that all arbitrary distributions can be approximated by a Coxian or Erlang distribution based on the coefficient of variation of the original distribution. The use of other phase-type distributions may increase the accuracy of the approximation when compared to empirical or known parametric distributions. This improved approximation technique should

enhance the accuracy of the lifetime estimation when compared to simulated results. Furthermore, modifications of this procedure that reduce the number of phases used to approximate each sojourn time distribution would greatly reduce computational effort due to a dimensionality reduction of the infinitesimal generator matrix and associated degradation rate matrix. In their recent work on phase-type distributions, Osogami and Harchol-Balter [22] present a moment-matching algorithm that nearly minimizes the number of phases used to approximate an arbitrary distribution. The time available for observation of the system's operating environment, and an acceptable balance between the accuracy of the phase-type approximations and the number of phases used for each state, will determine the efficacy of the procedures developed in this thesis.

Bibliography

1. Abate, J. and W. Whitt (1995). Numerical inversion of Laplace transforms of probability distributions. *ORSA Journal on Computing*, **7**, 36-43.
2. Abdel-Hameed, M. (1984). Life distribution properties of devices subject to a Lévy wear process. *Mathematics of Operations Research*, **9** (4), 606-614.
3. Ahmad, M. and A. Sheikh (1984). Bernstein reliability model: derivation and estimation of parameters. *Reliability Engineering*, **8**, 131-148.
4. Altiok, T. (1985). On the phase-type approximations of general distributions. *IIE Transactions*, **17** (2), 110-116.
5. Bhattacharyya, G.K. and A. Fries (1982). Fatigue-failure models - Birnbaum-Saunders vs. inverse Gaussian. *IEEE Reliability*, **R-31**, 439-440.
6. Boucherie, R.J. and N.M. Van Dijk (2000). On a queueing network model for cellular mobile telecommunications networks. *Operations Research*, **48** (1), 38-49.
7. Chan, V. and W.Q. Meeker (2003). Estimation of degradation-based reliability in outdoor environments. Preprint.
8. Chao, M. (1999). Degradation analysis and related topics: some thoughts and a review. *Proc. Natl. Sci. Counc. ROC(A)*,**23** (5), 555-566.
9. Chinnam, R. (1999). On-line reliability estimation of individual components using degradation signals. *IEEE Transactions on Reliability*, **48** (6), 403-412.
10. Çinlar, E. (1977). Shock and wear models and Markov additive processes, in: Shimi, I.N. and C.P. Tsokos (Eds.), *The Theory and Applications of Reliability*. Academic Press, New York, 1977, 193-214.
11. Crk, V. (2000). Reliability assessment from degradation data. *2000 Proceedings Annual Reliability and Maintainability Symposium*, 155-161.
12. Esary, J.D., A.W. Marshall, and F. Proschan (1973). Shock models and wear processes. *Annals of Probability*, **1** (4), 627-649.
13. Fang, Y. and I. Chlamtac (2002). Analytical generalized results for handoff probability in wireless networks. *IEEE Transactions on Communications*, Vol. **50**, No. **3**, 396-399.
14. Gertsbakh, I.B. and K.B Kordonsky (1969). *Models of Failure*. Springer-Verlag, New York.
15. Jayasuriya, A, D. Green, and J. Asenstorfer (2001). Modelling service time distribution in cellular networks using phase-type service distributions. *IEEE International Conference on Communication, 2001*, **2**, 11-14.

16. Kharoufeh, J.P. (2003). Explicit results for wear processes in a Markovian environment. *Operations Research Letters*, **31**, 237-244.

17. Kharoufeh, J.P. and J.A. Sipe (2004). Evaluating failure time probabilities for a Markovian wear process. Forthcoming in *Computers and Operations Research*.

18. Lawless, J. (2003). Degradation models and failure prediction. *International Conference in Reliability and Survival Analysis (ICRSA)*, University of South Carolina, Columbia SC, May 21-24, 2003.

19. Limnios, N. and G. Oprisan (1993). A unified approach for reliability and performability evaluation of semi-Markov systems. *Applied Stochastic Models in Business and Industry*, **15**, 353-368.

20. Lu, J. and W. Meeker (1993). Using degradation measures to estimate a time-to-failure distribution. *Technometrics*, **35** (2), 161-174.

21. Neuts, M.F. (1981). *Matrix-Geometric Solutions in Stochastic Models - An Algorithmic Approach*. John Hopkins University Press.

22. Osogami, T. and M. Harchol-Balter (2003). A closed-form solution for mapping general distributions to minimal PH distributions. *12th International Conference on Modelling Tools and Techniques for Computer and Communication System Performance Evaluation (TOOLS 2003)*, 200-217.

23. Perros, H. (1994). *Queueing Networks with Blocking*. Oxford University Press, New York.

24. Ro, C.W. and K.S. Trivedi (1997). Performability analysis of handoff calls in personal communication networks. *Proceedings of the Sixth International Conference on Communications and Networks*, 116-121.

25. Singpurwalla, N.D. (1995). Survival in dynamic environments. *Statistical Science*, **10** (1), 86-103.

26. Vidalis, M.I. and H.T. Papadopoulos (1999). Markovian analysis of production lines with Coxian-2 service times. *International Transactions in Operations Research*, **6**, 495-524.

27. Vysokovskii, E.S. (1966). Reliability of tools used on semi-automatic lathes. *Russian Engineering Journal*, **XLVI** (6), 46-50.

28. Wu, S. and T. Tsai (2000). Estimation of time-to-failure distribution derived from a degradation model using fuzzy clustering. *Quality and Reliability Engineering International*, **16**, 261-267.

29. Yacout, A., S. Salvatores, and Y. Orechwa (1996). Degradation analysis estimates of the time-to-failure distribution of irradiated fuel elements. *Nuclear Technology*, **113**, 177-188.

Appendix A. Code for Brake Pad Example

```
%************************************************************************
% Plot_SMP_weib.m
%
% This MATLAB program runs the semi-Markov process (SMP) simulation
(semi_Markov_weib.m) and then the 'simulation plus phase-type approximations
plus analytical solution' program SMPweibstates.m.  The CDF values are then
compared in a plot (Figure 2).
%
%************************************************************************
%
% Author:  Captain Christopher J. Solo, M.S. candidate, Operations Research
%          Air Force Institute of Technology
%   Date:  January 29, 2004
%
%************************************************************************
%
clear all;
%
t=0.005:0.05:50.0;  % vector of time points at which to evaluate both CDFs
%
semi_Markov_weib;   % calls program to simulate semi-Markov process (SMP) and
obtain
%                     failure times
%
SMPweibstates;   % calls program to simulate SMP (long term), compute phase-type
%                  approximations, and produce analytical results of failure
                   time distribution
%
% F = cdf values from simulation
% FailureTimeProb = cdf values from phase-type analytical solution
%
% Produces plot of simulated failure time distribution vs. analytical
% CDF derived via conversion of environment process from SMP to CTMC
%
figure(2);
plot(t,F,'r');
hold on;
plot(t,FailureTimeProb,'b');
grid off;
title('Failure-time CDF values:  Simulated vs. Analytical');
xlabel('t');
ylabel('P(T < t)');
legend('Simulated CDF','Analytical CDF',0);
```

```
%*****************************************************************************
% PROGRAM semi_Markov_weib.m
%
% The purpose of this MATLAB program is to simulate a finite-state semi-Markov
% process.  The process is simulated in order to validate analytical solutions
% for the distribution and moments of the lifetime of components whose
% degradation threshold is WearThreshold.  The program uses function "rando" in
% order to select the next state after a state transition.
%
%  Orig Author: Jeffrey P. Kharoufeh, Ph.D. candidate, IE & OR,
%               Penn State University
%         Date: January 15, 2001
%   Revised by: Captain Chris Solo, M.S. candidate, OR,
%               Air Force Institute of Technology
%         Date: January 29, 2004
%
%*****************************************************************************
% VARIABLE DEFINTIONS
*****************************************************************
%
% WearThreshold  = Degradation threshold (signifies failure)
% k  = an index variable
%*****************************************************************************

% VECTOR/FUNCTION DEFINITIONS
%*********************************************************
%
% Lifetime = Vector of lifetimes
% B = Vector for the initial probability distribution of the Markov process
% P = Probability transition matrix
% V = Vector of degradation rates (i.e., V(i)= degradation rate when environment
%     is in state i).
% Z = Vector of environment states
%*****************************************************************************

% The program assumes a state space of the form S={1,2,...K}

semi_Markov_input_weib;        % Obtain intialization parameters
Z = [];                        % Initialize Z.
for k = 1:length(Lifetime)
   Z = [];
   Z(1)=rando(B);   % Initial state of the environment at time 0
   Lifetime(k) = weibrnd(K(Z(1),1),K(Z(1),2)); % Time spent in initial state
   D = 0.0;                                    % At time 0, degradation is 0
   D = D + Lifetime(k)*V(Z(1));   % Add degradation to cumulative degradation
   i=1;
   while D < WearThreshold % Do while cumulative degradation < WearThreshold
      Z(i+1) = rando(P(Z(i),:));  % Use the matrix P to determine next state
      new_time = weibrnd(K(Z(i+1),1),K(Z(i+1),2)); % Time spent in state i+1
      D = D + new_time*V(Z(i+1)); % Update cumulative degradation
      Lifetime(k) = Lifetime(k) + new_time;     % Update total lifetime
      i=i+1;
   end
Lifetime(k)=Lifetime(k)-(1/V(Z(i)))*(D-WearThreshold); % Subtract time after
                                                       % reaching threshold
end
get_cdf_weib; % Computes empirical distribution
```

```
%*************************************************************************
% PROGRAM semi_Markov_input_weib.m
%
% The purpose of this MATLAB program is to provide input data to program
% semi_Markov_weib.m for simulation of a finite-state semi-Markov process
% (K states).  The process is simulated in order to validate analytical
% solutions for the distribution and moments of the failure time of systems with
% degradation threshold L subject to a SMP environment.  The program uses
% function "rando" in order to select the next state after a state transition.
%
%  Orig Author:  Jeffrey P. Kharoufeh, Ph.D. candidate, IE & OR,
%                Penn State University
%         Date:  January 15, 2001
%   Revised by:  Captain Christopher Solo, M.S., OR,
%                Air Force Institute of Technology
% Last Revised:  January 17, 2004
%
%*************************************************************************
% VARIABLE DEFINTIONS
*************************************************************
%
% WearThreshold  = degradation threshold (system fails after accumulating L
%                  amount of damage)
%
%*************************************************************************
% VECTOR/FUNCTION DEFINITIONS
%*********************************************************
%
% T = Vector of failure time observations
% B = Vector for the initial probability distribution of the semi-Markov process
% P = Probability transition matrix
% V = Vector of degradation rates (i.e., V(i)= degrad. rate when environment is
%     in state i).
%*************************************************************************

% Initialize variable and vector values

WearThreshold = 3.0;  % Component reaches 'failure' with this level of
degradation
N = 3;                % number of states

Lifetime = zeros(1,100000);   % number of simulations (failure time
observations)
B = [1 zeros(1,N-1)];
%
S = [1,2,3];               % states of the environment
V = [1,10,20];      % vector of degradation rates
%
P = [0.0 0.7 0.3;   % transition probability matrix;
     0.6 0.0 0.4;
     0.2 0.8 0.0];

K = [4 2;  % Parameter matrix--each row gives parameters of distribution
     5 3;
     6 4];
```

```
%*************************************************************************
% PROGRAM get_cdf_weib.m
%
% The purpose of this MATLAB program is to construct an empirical distribution
% function based upon a vector of observations of some continuous random
% variable.  The values of the CDF are generally used for the purpose of
% performing hypothesis tests on the equality of two nonparametric distributions.
% Other approaches for testing may be the Cramer von Mises test, which is
% similar to the K-S two-sample test, but slightly more robust.
%
%  Orig Author:  Jeffrey P. Kharoufeh, Ph.D. candidate, IE & OR,
%                Penn State University
%          Date:  June 2000
%   Revised by:  Captain Chris Solo, M.S. candidate, OR,
%                Air Force Institute of Technology
% Last Revised:  January 18, 2004
%*************************************************************************
% VECTOR DEFINITIONS
%*********************************************************************
%
% Lifetime = The finite-dimensional vector of real observations (read from
%            semi_Markov_weib.m)
% t = The vector of points at which to estimate the CDF.
% F = The vector of CDF estimates.  That is, F(i) = F(t(i)), i=1,2,...,n of
%     {1,2,...,C}.
%*************************************************************************
%
F = zeros(1,length(t));  % This ensures that t and F are vectors of equal length
%
% Compute the cdf value at the point t0
%
for i = 1:length(t)
    F(i) = 0.0;
    for j = 1:length(Lifetime)
        if Lifetime(j)<= t(i)
            F(i) = F(i) + 1/length(Lifetime);
        else
            F(i) = F(i);
        end
    end
end
%
% WRITE OUTPUTS TO A TAB DELIMITED TEXT FILE
%
fid = fopen('F:\AFIT\Thesis\Solo Thesis\SMP
simulation\results\brakesimulated.out','w');
for b=1:length(t)
   fprintf(fid, '%4.6f\t%4.6f\n',t(b),F(b));
end
fclose(fid)
```

```
%************************************************************************
% SMPweibstates.m
%
% The purpose of this MATLAB program is to simulate a semi-Markov process (SMP)
% on a state space S and convert it into a continuous-time Markov chain, using a
% phase-type distribution to approximate the holding time distribution of each
% environment state.  Each of the environment states is known to have a Weibull
% holding time distribution (each with different parameters.)  The output
% is the analytical solution for the failure time distribution of a system
% subject to the SMP environment, based on the results of Kharoufeh (2003) and
% Kharoufeh and Sipe (2004).
%
% Original code for process.m and rando.m obtained from Craig L. Zirbel at
% http://www-math.bgsu.edu/~zirbel/ap/  It appeared that the code was based
% on a discrete time Markov chain, and Captain Steve Cox modified it to work
% for a continuous-time Markov chain.  Captain Chris Solo modified it to be
% a SMP (January 2004).
%
%************************************************************************
%
%       Author:  Captain Christopher J. Solo, M.S. candidate,
%                Operations Research, Air Force Institute of Technology
%         Date:  September 13, 2003
% Last Revised:  January 18, 2004
%
%************************************************************************
%                              USER INPUTS
% Tmax = time of SMP run
% numruns = number of simulations (runs)
% mu = initial probability vector of environment states
% S = state space vector
% R = degradation rate vector of environment states
% P = transition probability matrix
% K = matrix of Weibull parameters for environment states
% M = large value used in Big M (hot potato) method
%
% Additionally, rando.m, phasetypeapprox.m, PHvalues.m, invt_lap_Solo.m, and
% failLST.m are needed.
%
%*************************************************************************
% For many short runs, set numruns high and Tmax low.  For one long run,
% set numruns equal to 1 and make Tmax high.

format long g;

%*********************************************************************
%                            User Inputs
%
numruns = 1;       % the number of times (runs) to simulate the process

Tmax = 10000;     % This is the length of the run of the environment process

mu = [1,0,0,0];    % Semi-Markov process (SMP) starts in state 1

S =  [1,2,3];    % There are three states defined

R =  [.1,1,2];  % Degradation rate of each state of environment
```

```
P = [0.0 0.7 0.3;   % transition probability matrix;
     0.6 0.0 0.4;
     0.2 0.8 0.0];

K = [4 2;          % Parameter matrix--each row gives parameters of different
distribution
     5 3;
     6 4];
%
%*****************************************************************
%
% Simulation of the semi-Markov process (SMP)
%
statetimeTij = zeros(length(S),length(S));
statetimeTi = [];
scale = 1;
%
TimeToFail(1) = 0;       % start times at 0
%Tall(runnum,1) = 0;
x(1) = rando(mu);        % determine starting state (time 0, not time 1)
%xall(runnum,1) = x(1);
rates(1) = R(x(1));
%ratesall(runnum,1) = rates(1);
i = 1;
%
while TimeToFail(i) < Tmax,
    x(i+1) = rando(P(x(i),:));              % Row vector from P-matrix
determines next transitions
    %xall(runnum,i+1) = x(i+1);
    time(i) = weibrnd(K(x(i),1),K(x(i),2)); % Generate Weibull rv based on
                                            % current state
    statetimeTij(x(i),x(i+1)) = statetimeTij(x(i),x(i+1)) + time(i);
    TimeToFail(i+1) = TimeToFail(i) + time(i);  % Add to current time
    %Tall(runnum,i+1) = T(i+1);
    rates(i+1) = R(x(i+1));     % Determine the rate needed for this transition
    %ratesall(runnum,i+1) = rates(i+1);
    i=i+1;
end
%
% Determine the number of transitions from one state to the next
%
numij = zeros(length(S),length(S));
for k = 1:length(x)-2
    numij(x(k),x(k+1)) = numij(x(k),x(k+1)) + 1;
end
%
% Compute the observed transition rates among the environment states
%
for i = 1:length(numij)
        for j = 1:length(numij)
            if j ~= i
                Qhat(i,j) = numij(i,j)/sum(statetimeTij(i,:));
            else
                Qhat(i,j) = 0;
            end
        end
```

```
        Qhat(i,i) = -sum(Qhat(i,:));
        statetimeTi(i,1) = sum(statetimeTij(i,:)); % total time spent in state i
    end
%
%*************************************************************************
% This section creates vectors of the holding times in States 1,2, and 3
%  ****IF ADDITIONAL STATES ARE ADDED, THEY MUST BE HARDCODED HERE****
%
for i = 1:length(time)
    for j = 1:length(S)
        if rates(i) == R(j)
           HoldingTime(i,j) = time(i);   % creates matrix where column i
        end                              % represents holding times in state i
    end
end
State1times = HoldingTime(:,1);
State1times = State1times(find(State1times)); % eliminates zeros in holding time
                                              % column
State2times = HoldingTime(:,2);
State2times = State2times(find(State2times)); % eliminates zeros in holding time
                                              % column
State3times = HoldingTime(:,3);
State3times = State3times(find(State3times)); % eliminates zeros in holding time
                                              % column
%
%*************************************************************************
% This section computes the phase-type approximations for each of the
% environment states
%    ****IF ADDITIONAL STATES ARE ADDED, THEY MUST BE HARDCODED HERE****
%
M=10000;     % this is the large value for the 'hot potato method'
             % i.e. since there is really no 'absorbing phase' for any
             % state, we wish to visit each absorbing phase
             % instantaneously.
             % therefore, its rate is extremely large (approaches infinity)
%

for State = 1:length(S)
    clear A;
    clear alpha1;
    clear alpha2;
    clear alpha;
    clear beta;
    clear k;
    clear T;
    clear To;
    if State == 1
        distribution = 2;
        A = State1times;
        alpha1=0; % parameter not needed; dummy value only
        alpha2=0; % parameter not needed; dummy value only
        alpha=K(State,2);
        beta=K(State,1);
        Aparam = 0; % parameter not needed; dummy value only
        Bparam = 0; % parameter not needed; dummy value only
        [k,T,To] =
phasetypeapproxmixed(A,alpha,beta,alpha1,alpha2,Aparam,Bparam,distribution);
```

```
% creates phase-type approximations to each of the holding time distributions
        k1=k;
        T1=T;
        To1=To;
        zerovector1=zeros(1,k1);
        Psimatrix11 = [ T1          To1;      % subgenerator matrix for state 1
                    zerovector1  Qhat(1,1)*M];
    elseif State == 2
        distribution = 2;
        A = State2times;
        alpha1=0; % parameter not needed; dummy value only
        alpha2=0; % parameter not needed; dummy value only
        alpha=K(State,2);
        beta=K(State,1);
        Aparam = 0; % parameter not needed; dummy value only
        Bparam = 0; % parameter not needed; dummy value only
        [k,T,To] =
phasetypeapproxmixed(A,alpha,beta,alpha1,alpha2,Aparam,Bparam,distribution);
% creates phase-type approximations to each of the holding time distributions
        k2=k;
        T2=T;
        To2=To;
        zerovector2=zeros(1,k2);
        Psimatrix22 = [  T2       To2;   % subgenerator matrix for state 2
                    zerovector2  Qhat(2,2)*M];
    elseif State == 3
        distribution = 2;
        A = State3times;
        alpha=K(State,2);
        beta=K(State,1);
        alpha1=0; % parameter not needed; dummy value only
        alpha2=0;  % parameter not needed; dummy value only
        Aparam = 0; % parameter not needed; dummy value only
        Bparam = 0; % parameter not needed; dummy value only
        [k,T,To] =
phasetypeapproxmixed(A,alpha,beta,alpha1,alpha2,Aparam,Bparam,distribution);
% creates phase-type approximations to each of the holding time distributions
        k3=k;
        T3=T;
        To3=To;
        zerovector3=zeros(1,k3);
        Psimatrix33 = [  T3     To3;      % subgenerator matrix for state 3
                  zerovector3  Qhat(3,3)*M];
    end
end
%
%***************************************************************************
% OFF-DIAGONAL subgenerator matrices
% ****IF ADDITIONAL STATES ARE ADDED, THEY MUST BE HARDCODED HERE****
%
Psimatrix12 = zeros(k1+1,k2+1);
Psimatrix12(k1+1,1)=Qhat(1,2)*M;
%
Psimatrix13 = zeros(k1+1,k3+1);
Psimatrix13(k1+1,1)=Qhat(1,3)*M;
%
Psimatrix21 = zeros(k2+1,k1+1);
```

```
Psimatrix21(k2+1,1)=Qhat(2,1)*M;
%
Psimatrix23 = zeros(k2+1,k3+1);
Psimatrix23(k2+1,1)=Qhat(2,3)*M;
%
Psimatrix31 = zeros(k3+1,k1+1);
Psimatrix31(k3+1,1)=Qhat(3,1)*M;
%
Psimatrix32 = zeros(k3+1,k2+1);
Psimatrix32(k3+1,1)=Qhat(3,2)*M;
%
Psimatrix = [Psimatrix11 Psimatrix12 Psimatrix13;   % overall generator matrix
             Psimatrix21 Psimatrix22 Psimatrix23;   % of (newly-created) CTMC
             Psimatrix31 Psimatrix32 Psimatrix33];
%
%************************************************************************
%      This section creates the expanded degradation rate matrix Lambda
%  ****IF ADDITIONAL STATES ARE ADDED, THEY MUST BE HARDCODED HERE****
%
V=diag(R);
L11 = diag(diag(V(1,1)*ones(k1+1)));
L12 = zeros(k1+1,k2+1);
L13 = zeros(k1+1,k3+1);
L21 = zeros(k2+1,k1+1);
L22 = diag(diag(V(2,2)*ones(k2+1)));
L23 = zeros(k2+1,k3+1);
L31 = zeros(k3+1,k1+1);
L32 = zeros(k3+1,k2+1);
L33 = diag(diag(V(3,3)*ones(k3+1)));
%
L = [L11 L12 L13;  % expanded degradation rate matrix
     L21 L22 L23;
     L31 L32 L33];
%
%************************************************************************
% Display the results of the phase-type approximations
%
fprintf(' Size of generator matrix (Psimatrix):  %g x %g \n', length(Psimatrix),
length(Psimatrix))
fprintf('\n')
fprintf(' Size of expanded degradation rate matrix (Lambda):  %g x %g \n',
length(L), length(L))
fprintf('\n')
fprintf(' ****************************** \n')
fprintf('\n')
%
%************************************************************************
% This section computes and plots the failure time distribution of the
% system exposed to the SMP environment
%
for w = 1:length(t)
    FailureTimeProb(w) = invt_lap_Solo(t(w),Psimatrix,L);
    if FailureTimeProb(w) > 1
        FailureTimeProb(w) = 1;   % eliminates CDF values > 1
    elseif FailureTimeProb(w) < 0
        FailureTimeProb(w) = 0;   % eliminates CDF values < 0
    end
```

```
end
%
%*************************************************************************
%  Compute first and second moment of failure time distribution
%
First_Moment = invt_lap_first_moment(WearThreshold,Psimatrix,L);
%
Second_Moment = invt_lap_second_moment(WearThreshold,Psimatrix,L);
%
%*************************************************************************
% WRITE OUTPUTS TO A TAB DELIMITED TEXT FILE
%
fid = fopen('f:\AFIT\Thesis\Solo Thesis\MATLAB files\SMP
simulation\results\brakeanalytical.out','w');
for k=1:length(t)
   fprintf(fid, '%4.6f\t%4.6f\n',t(k),FailureTimeProb(k));
end
fclose(fid);
```

Appendix B. Code for Turbine Blade Example

```
%************************************************************************
% Plot_SMP_mixed.m
%
% This MATLAB program runs the semi-Markov process (SMP) simulation
% (semi_Markov.m) and then the 'simulation plus phase-type approximations plus
% analytical solution' program SMPmixedstates.m.  The CDF values are then
% compared in a plot (Figure 2).
%
%************************************************************************
%
% Author:  Captain Christopher J. Solo, M.S. candidate, Operations Research
%          Air Force Institute of Technology
%   Date:  January 29, 2004
%
%************************************************************************
%
clear all;
%
t=0.005:0.05:50.0;  % vector of time points at which to evaluate both CDFs
%
semi_Markov_mixed;  % calls program to simulate semi-Markov process (SMP) and
obtain
%                     failure times
%
% SMPmixedstates;   % calls program to simulate SMP (long term), compute phase-
                    % type approximations, and produce analytical results of
                    % failure time distribution
%
% F = cdf values from simulation
% FailureTimeProb = cdf values from phase-type analytical solution
%
% Produces plot of simulated failure-time distribution vs. analytical
% CDF derived via conversion of environment process from SMP to CTMC
%
 figure(2);
 plot(t,F,'r');
 hold on;
 plot(t,FailureTimeProb,'b');
 grid off;
 title('Failure-time CDF values:  Simulated vs. Analytical');
 xlabel('t');
 ylabel('P(T < t)');
 legend('Simulated CDF','Analytical CDF',0);
```

```
%*************************************************************************
% PROGRAM semi_Markov_mixed.m
%
% The purpose of this MATLAB program is to simulate a finite-state semi-Markov
% process.  The process is simulated in order to validate analytical solutions
% for the distribution and moments of the lifetime of components whose
% degradation threshold is WearThreshold.  The program uses function "rando" in
% order to select the next state after a state transition.
%
%  Orig Author: Jeffrey P. Kharoufeh, Ph.D. candidate, IE & OR,
%               Penn State University
%         Date: January 15, 2001
%   Revised by: Captain Chris Solo, M.S. candidate, OR, \
%               Air Force Institute of Technology
%         Date: 29 January 2004
%
%*************************************************************************
% VARIABLE DEFINTIONS
*****************************************************************
%
% WearThreshold  = Degradation threshold (signifies failure)
% k  = an index variable
%*************************************************************************
% VECTOR/FUNCTION DEFINITIONS
%*******************************************************
%
% Lifetime = Vector of lifetimes
% B = Vector for the initial probability distribution of the Markov process
% P = Probability transition matrix
% V = Vector of degradation rates (i.e., V(i)= degradation rate when environment
%     is in state i).
% Z = Vector of environment states
%*************************************************************************

% The program assumes a state space of the form S={1,2,...K}

semi_Markov_input_mixed;             % Obtain intialization parameters
Z = [];                              % Initialize Z.

for k = 1:length(Lifetime)
   Z = [];
   Z(1)=rando(B);              % Initial state of the environment at time 0

   if (Z(1) == S(1)) | (Z(1) == S(2))
       Lifetime(k) = betarnd(K(Z(1),1),K(Z(1),2)); % Time spent in initial state
   elseif (Z(1) == S(3)) | (Z(1) == S(4))
       Lifetime(k) = weibrnd(K(Z(1),1),K(Z(1),2)); % Time spent in initial state
   end

   D = 0.0;                                  % At time 0, degradation is 0
   D = D + Lifetime(k)*V(Z(1));   % Add degradation to cumulative degradation
   i=1;
   while D < WearThreshold % Do while cumulative degradation is < WearThreshold
      Z(i+1) = rando(P(Z(i),:));  % Use the matrix P to determine next state
      if (Z(i+1) == S(1)) | (Z(i+1) == S(2))
         new_time = betarnd(K(Z(i+1),1),K(Z(i+1),2)); % Time spent in state i+1
      elseif (Z(i+1) == S(3)) | (Z(i+1) == S(4))
```

```
         new_time = weibrnd(K(Z(i+1),1),K(Z(i+1),2)); % Time spent in state i+1
      end
      D = D + new_time*V(Z(i+1)); % Update cumulative degradation
      Lifetime(k) = Lifetime(k) + new_time;     % Update total lifetime
      i=i+1;
   end
   Lifetime(k)=Lifetime(k)-(1/V(Z(i)))*(D-WearThreshold); % Subtract time after
                                                          % reaching threshold
end
get_cdf_mixed; % Computes empirical distribution
```

```
%*************************************************************************
% PROGRAM semi_Markov_input_mixed.m
%
% The purpose of this MATLAB program is to provide input data to program
% semi_Markov_mixed.m for simulation of a finite-state semi-Markov process (K
% states).  The process is simulated in order to validate analytical solutions
% for the distribution and moments of the failure time of systems with
% degradation threshold L subject to a SMP environment.  The program uses
% function "rando" in order to select the next state after a state transition.
%
%  Orig Author:  Jeffrey P. Kharoufeh, Ph.D. candidate, IE & OR,
%                Penn State University
%         Date:  January 15, 2001
%   Revised by:  Captain Christopher Solo, M.S., OR,
%                Air Force Institute of Technology
% Last Revised:  January 17, 2004
%*************************************************************************
% VARIABLE DEFINTIONS
*****************************************************************
%
% WearThreshold  = degradation threshold (system fails after accumulating this
% amount of damage)
%
%*************************************************************************
% VECTOR/FUNCTION DEFINITIONS
%*********************************************************
%
% T = Vector of failure time observations
% B = Vector for the initial probability distribution of the semi-Markov process
% P = Probability transition matrix
% V = Vector of degradation rates (i.e., V(i)= degrad. rate when environment is
%     in state i).
%*************************************************************************
% Initialize variable and vector values

WearThreshold = 20.0;  % Component reaches 'failure' with this level of
                       % degradation
N = 4;                 % number of states

Lifetime = zeros(1,100000);   % number of simulations (failure time obs)
B = [1 zeros(1,N-1)];
%
S = [1,2,3,4];               % states of the environment
V = [1.7/10, 4.3/10, 7.5/10, 12.9/10];      % vector of degradation rates
%
P = [0 .6 .2 .2;      % transition probability matrix;
    .5  0 .4 .1;
    .2 .4  0 .4;
    .1 .3 .6  0];
%
K = [3 5;     % Parameter matrix--each row gives parameters of distribution
     2 4;
     5 2;
     6 3];
```

```
%*************************************************************************
% PROGRAM get_cdf_mixed.m
%
% The purpose of this MATLAB program is to construct an empirical distribution
% function based upon a vector of observations of some continuous random
% variable.  The values of the CDF are generally used for the purpose of
% performing hypothesis test on the equality of two nonparametric distributions.
% Other approaches for testing may be the Cramer von Mises test, which is
% similar to the K-S two-sample test, but slightly more robust.
%
%  Orig Author:  Jeffrey P. Kharoufeh, Ph.D. candidate, IE & OR,
%                Penn State University
%         Date:  June 2000
%   Revised by:  Captain Chris Solo, M.S. candidate, OR,
%                Air Force Institute of Technology
% Last Revised:  January 18, 2004
%
%*************************************************************************
% VECTOR DEFINITIONS
%*********************************************************************
%
% Lifetime = The finite-dimensional vector of real obs
% t = The vector of points at which to estimate the CDF.
% F = The vector of CDF estimates.  That is, F(i) = F(t(i)), i=1,2,...,nof
%     {1,2,...,C}.
%*************************************************************************
%
F = zeros(1,length(t));  % This ensures that t and F are vectors of equal length
%
% Compute the cdf value at the point t0
%
for i = 1:length(t)
    F(i) = 0.0;
    for j = 1:length(Lifetime)
        if Lifetime(j)<= t(i)
            F(i) = F(i) + 1/length(Lifetime);
        else
            F(i) = F(i);
        end
    end
end
%
% WRITE OUTPUTS TO A TAB DELIMITED TEXT FILE
%
fid = fopen('F:\AFIT\Thesis\Solo Thesis\SMP
simulation\results\turbinesimulated.out','w');
for b=1:length(t)
   fprintf(fid, '%4.6f\t%4.6f\n',t(b),F(b));
end
fclose(fid)
```

```
%************************************************************************
% SMPmixedstates.m
%
% The purpose of this MATLAB program is to simulate a semi-Markov process (SMP)
% on a state space S and convert it into a continuous-time Markov chain, using a
% phase-type distribution to approximate the holding time distribution of each
% environment state.  Each of the environment states is known to have a Weibull
% holding time distribution (each with different parameters.)  The output
% is the analytical solution for the failure-time distribution of a system
% subject to the SMP environment, based on the results of Kharoufeh (2003) and
% Kharoufeh and Sipe (2004).
%
% Original code for process.m and rando.m obtained from Craig L. Zirbel at
% http://www-math.bgsu.edu/~zirbel/ap/  It appeared that the code was based
% on a discrete time Markov chain, and Captain Steve Cox modified it to work
% for a continuous-time Markov chain.  Captain Chris Solo modified it to be
% a SMP (January 2004).
%
%************************************************************************
%
%       Author:  Captain Christopher J. Solo, M.S. candidate,
%                Operations Research
%                Air Force Institute of Technology
%         Date:  September 13, 2003
% Last Revised:  January 18, 2004
%
%************************************************************************
%                              USER INPUTS
% Tmax = time of SMP run
% numruns = number of simulations (runs)
% mu = initial probability vector of environment states
% S = state space vector
% R = degradation rate vector of environment states
% P = transition probability matrix
% K = matrix of Weibull parameters for environment states
% M = large value used in Big M (hot potato) method
%
% Additionally, rando.m, phasetypeapproxmixed.m, PHvalues.m, invt_lap_Solo.m,
% and failLST.m are needed.
%
%************************************************************************
% For many short runs, set numruns high and Tmax low.  For one long run,
% set numruns equal to 1 and make Tmax high.

format long g;

%*********************************************************************
%                            User Inputs
%
numruns = 1;       % the number of times (runs) to simulate the process

Tmax = 10000;     % This is the length of the run of the environment process

mu = [1,0,0,0];    % Semi-Markov process (SMP) starts in state 1

S =  [1,2,3,4];    % There are three states defined
```

```
R =  [1.7/10, 4.3/10, 7.5/10, 12.9/10];  % Degradation rate of each state of
                                         % environment

P = [0 .6 .2 .2;   % transition probability matrix;
    .5  0 .4 .1;
    .2 .4  0 .4;
    .1 .3 .6  0];

K = [3 5;      % Parameter matrix--each row gives parameters of distribution
     2 4;
     5 2;
     6 3];
%
%*****************************************************************
%
% Simulation of the semi-Markov process (SMP)
%
statetimeTij = zeros(length(S),length(S));
statetimeTi = [];
scale = 1;
%
TimeToFail(1) = 0;       % start times at 0
%Tall(runnum,1) = 0;
x(1) = rando(mu);        % determine starting state (time 0, not time 1)
%xall(runnum,1) = x(1);
rates(1) = R(x(1));
%ratesall(runnum,1) = rates(1);
i = 1;
%
while TimeToFail(i) < Tmax,
    x(i+1) = rando(P(x(i),:)); % Row vector from P-matrix determines next
                               % transitions
    %xall(runnum,i+1) = x(i+1);
    if x(i) == S(1) |  x(i) == S(2)
        time(i) = betarnd(K(x(i),1),K(x(i),2)); % Generate beta rv based on
                                                % current state
    elseif x(i) == S(3) |  x(i) == S(4)
        time(i) = weibrnd(K(x(i),1),K(x(i),2)); % Generate Weibull rv based on
                                                % current state
    end
    statetimeTij(x(i),x(i+1)) = statetimeTij(x(i),x(i+1)) + time(i);
    TimeToFail(i+1) = TimeToFail(i) + time(i);  % Add to current time
    %Tall(runnum,i+1) = T(i+1);
    rates(i+1) = R(x(i+1));      % Determine the rate needed for this transition
    %ratesall(runnum,i+1) = rates(i+1);
    i=i+1;
end
%
% Determine the number of transitions from one state to the next
%
numij = zeros(length(S),length(S));
for k = 1:length(x)-2
    numij(x(k),x(k+1)) = numij(x(k),x(k+1)) + 1;
end
%
% Compute the observed transition rates among the environment states
%
```

```
for i = 1:length(numij)
        for j = 1:length(numij)
            if j ~= i
                Qhat(i,j) = numij(i,j)/sum(statetimeTij(i,:));
            else
                Qhat(i,j) = 0;
            end
        end
        Qhat(i,i) = -sum(Qhat(i,:));
        statetimeTi(i,1) = sum(statetimeTij(i,:)); % total time spent in state i
    end
%
%************************************************************************
% This section creates vectors of the holding times in States 1,2, and 3
%  ****IF ADDITIONAL STATES ARE ADDED, THEY MUST BE HARDCODED HERE****
%
for i = 1:length(time)
    for j = 1:length(S)
        if rates(i) == R(j)
           HoldingTime(i,j) = time(i);   % creates matrix where column i
        end                               % represents holding times in state i
    end
end
State1times = HoldingTime(:,1);
State1times = State1times(find(State1times)); % eliminates zeros in holding time
                                              % column
State2times = HoldingTime(:,2);
State2times = State2times(find(State2times)); % eliminates zeros in holding time
                                              % column
State3times = HoldingTime(:,3);
State3times = State3times(find(State3times)); % eliminates zeros in holding time
                                              % column
State4times = HoldingTime(:,4);
State4times = State4times(find(State4times)); % eliminates zeros in holding time
                                              % column
%
%************************************************************************
% This section computes the phase-type approximations for each of the
% environment states
%    ****IF ADDITIONAL STATES ARE ADDED, THEY MUST BE HARDCODED HERE****
%
M=10000;     % this is the large value for the 'hot potato method'
             % i.e. since there is really no 'absorbing phase' for any
             % state, we wish to visit each absorbing phase
             % instantaneously.
             % therefore, its rate is extremely large (approaches infinity)
%

for State = 1:length(S)
    clear A;
    clear alpha1;
    clear alpha2;
    clear alpha;
    clear beta;
    clear k;
    clear T;
    clear To;
```

```
    if State == 1
        distribution = 3;
        A = State1times;
        alpha1=K(State,1);
        alpha2=K(State,2);
        alpha=0; % parameter not needed; dummy value only
        beta=0;  % parameter not needed; dummy value only
        Aparam=0; % parameter not needed; dummy value only
        Bparam=0; % parameter not needed; dummy value only
        [k,T,To] =
phasetypeapproxmixed(A,alpha,beta,alpha1,alpha2,Aparam,Bparam,distribution);
% creates phase-type approximations to each of the holding time distributions
        k1=k;
        T1=T;
        To1=To;
        zerovector1=zeros(1,k1);
        Psimatrix11 = [   T1     To1;  % subgenerator matrix for state 1
                      zerovector1  Qhat(1,1)*M];
    elseif State == 2
        distribution = 3;
        A = State2times;
        alpha1=K(State,1);
        alpha2=K(State,2);
        alpha=0; % parameter not needed; dummy value only
        beta=0;  % parameter not needed; dummy value only
        Aparam=0; % parameter not needed; dummy value only
        Bparam=0; % parameter not needed; dummy value only
        [k,T,To] =
phasetypeapproxmixed(A,alpha,beta,alpha1,alpha2,Aparam,Bparam,distribution);
% creates phase-type approximations to each of the holding time distributions
        k2=k;
        T2=T;
        To2=To;
        zerovector2=zeros(1,k2);
        Psimatrix22 = [     T2    To2;      % subgenerator matrix for state 2
                    zerovector2  Qhat(2,2)*M];
    elseif State == 3
        distribution = 2;
        A = State3times;
        alpha=K(State,2);
        beta=K(State,1);
        alpha1=0; % parameter not needed; dummy value only
        alpha2=0;  % parameter not needed; dummy value only
        Aparam=0; % parameter not needed; dummy value only
        Bparam=0; % parameter not needed; dummy value only
        [k,T,To] =
phasetypeapproxmixed(A,alpha,beta,alpha1,alpha2,Aparam,Bparam,distribution);
% creates phase-type approximations to each of the holding time distributions
        k3=k;
        T3=T;
        To3=To;
        zerovector3=zeros(1,k3);
        Psimatrix33 = [   T3     To3;      % subgenerator matrix for state 3
                  zerovector3  Qhat(3,3)*M];
    elseif State == 4
        distribution = 2;
        A = State4times;
```

```
        alpha=K(State,2);
        beta=K(State,1);
        alpha1=0; % parameter not needed; dummy value only
        alpha2=0;  % parameter not needed; dummy value only
        Aparam=0; % parameter not needed; dummy value only
        Bparam=0; % parameter not needed; dummy value only
        [k,T,To] =
phasetypeapproxmixed(A,alpha,beta,alpha1,alpha2,Aparam,Bparam,distribution);
% creates phase-type approximations to each of the holding time distributions
        k4=k;
        T4=T;
        To4=To;
        zerovector4=zeros(1,k4);
        Psimatrix44 = [     T4      To4;      % subgenerator matrix for state 3
                    zerovector4  Qhat(4,4)*M];
    end
end
%
%************************************************************************
% OFF-DIAGONAL subgenerator matrices
% ****IF ADDITIONAL STATES ARE ADDED, THEY MUST BE HARDCODED HERE****
%
Psimatrix12 = zeros(k1+1,k2+1);
Psimatrix12(k1+1,1)=Qhat(1,2)*M;
%
Psimatrix13 = zeros(k1+1,k3+1);
Psimatrix13(k1+1,1)=Qhat(1,3)*M;
%
Psimatrix14 = zeros(k1+1,k4+1);
Psimatrix14(k1+1,1)=Qhat(1,4)*M;
%
Psimatrix21 = zeros(k2+1,k1+1);
Psimatrix21(k2+1,1)=Qhat(2,1)*M;
%
Psimatrix23 = zeros(k2+1,k3+1);
Psimatrix23(k2+1,1)=Qhat(2,3)*M;
%
Psimatrix24 = zeros(k2+1,k4+1);
Psimatrix24(k2+1,1)=Qhat(2,4)*M;
%
Psimatrix31 = zeros(k3+1,k1+1);
Psimatrix31(k3+1,1)=Qhat(3,1)*M;
%
Psimatrix32 = zeros(k3+1,k2+1);
Psimatrix32(k3+1,1)=Qhat(3,2)*M;
%
Psimatrix34 = zeros(k3+1,k4+1);
Psimatrix34(k3+1,1)=Qhat(3,4)*M;
%
Psimatrix41 = zeros(k4+1,k1+1);
Psimatrix41(k4+1,1)=Qhat(4,1)*M;
%
Psimatrix42 = zeros(k4+1,k2+1);
Psimatrix42(k4+1,1)=Qhat(4,2)*M;
%
Psimatrix43 = zeros(k4+1,k3+1);
Psimatrix43(k4+1,1)=Qhat(4,3)*M;
```

```
Psimatrix = [Psimatrix11 Psimatrix12 Psimatrix13 Psimatrix14;  % overall
             Psimatrix21 Psimatrix22 Psimatrix23 Psimatrix24;  % gener. matrix
             Psimatrix31 Psimatrix32 Psimatrix33 Psimatrix34;  % of (newly-
             Psimatrix41 Psimatrix42 Psimatrix43 Psimatrix44]; % created) CTMC
%
%*************************************************************************
%      This section creates the expanded degradation rate matrix Lambda
%  ****IF ADDITIONAL STATES ARE ADDED, THEY MUST BE HARDCODED HERE****
%
V=diag(R);
L11 = diag(diag(V(1,1)*ones(k1+1)));
L12 = zeros(k1+1,k2+1);
L13 = zeros(k1+1,k3+1);
L14 = zeros(k1+1,k4+1);
L21 = zeros(k2+1,k1+1);
L22 = diag(diag(V(2,2)*ones(k2+1)));
L23 = zeros(k2+1,k3+1);
L24 = zeros(k2+1,k4+1);
L31 = zeros(k3+1,k1+1);
L32 = zeros(k3+1,k2+1);
L33 = diag(diag(V(3,3)*ones(k3+1)));
L34 = zeros(k3+1,k4+1);
L41 = zeros(k4+1,k1+1);
L42 = zeros(k4+1,k2+1);
L43 = zeros(k4+1,k3+1);
L44 = diag(diag(V(4,4)*ones(k4+1)));
%
L = [L11 L12 L13 L14;  % expanded degradation rate matrix
     L21 L22 L23 L24;
     L31 L32 L33 L34;
     L41 L42 L43 L44];
%
%*************************************************************************
% Display the results of the phase-type approximations
%
fprintf(' Size of generator matrix (Psimatrix):  %g x %g \n', length(Psimatrix),
length(Psimatrix))
fprintf('\n')
fprintf(' Size of expanded degradation rate matrix (Lambda):  %g x %g \n',
length(L), length(L))
fprintf('\n')
fprintf(' ***************************** \n')
fprintf('\n')
%
%*************************************************************************
% This section computes and plots the failure time distribution of the
% system exposed to the SMP environment
%
for w = 1:length(t)
    FailureTimeProb(w) = invt_lap_Solo(t(w),Psimatrix,L);
    if FailureTimeProb(w) > 1
        FailureTimeProb(w) = 1;   % eliminates CDF values > 1
    elseif FailureTimeProb(w) < 0
        FailureTimeProb(w) = 0;   % eliminates CDF values < 0
    end
end
```

```
%*************************************************************************
%  Compute first and second moment of failure time distribution
%
First_Moment = invt_lap_first_moment(WearThreshold,Psimatrix,L);
%
Second_Moment = invt_lap_second_moment(WearThreshold,Psimatrix,L);
%
%*************************************************************************
%
% WRITE OUTPUTS TO A TAB DELIMITED TEXT FILE
%
fid = fopen('F:\AFIT\Thesis\Solo Thesis\SMP
simulation\results\turbineanalytical.out','w');
for k=1:length(t)
   fprintf(fid, '%4.6f\t%4.6f\n',t(k),FailureTimeProb(k));
end
fclose(fid);
```

Appendix C. Code for Coating Example

```
%************************************************************************
% Plot_SMP_five.m
%
% This MATLAB program runs the semi-Markov process (SMP) simulation
% (semi_Markov_five.m) and then the 'simulation plus phase-type approximations
% plus analytical solution' program SMPfivestates.m.  The CDF values are then
% compared in a plot (Figure 2).
%
%************************************************************************
%
% Author:  Captain Christopher J. Solo, M.S. candidate, Operations Research
%          Air Force Institute of Technology
%   Date:  January 29, 2004
%
%************************************************************************
%
clear all;
%
t=0.05:0.005:10;  % vector of time points at which to evaluate both CDFs
%
semi_Markov_five;   % calls program to simulate semi-Markov process (SMP) and
obtain
%                     failure times
%
SMPfivestates;   % calls program to simulate SMP (long term), compute phase-type
                 % approximations, & produce analytical results of failure time
                 % distribution
%
% F = cdf values from simulation
% FailureTimeProb = cdf values from phase-type analytical solution
%
% Produces plot of simulated failure-time distribution vs. analytical
% CDF derived via conversion of environment process from SMP to CTMC
%
figure(2);
plot(t,F,'r');
hold on;
plot(t,FailureTimeProb,'b');
grid off;
title('Failure-time CDF values:  Simulated vs. Analytical');
xlabel('t');
ylabel('P(T < t)');
legend('Simulated CDF','Analytical CDF',0);
```

```
%*************************************************************************
% PROGRAM semi_Markov_five.m
%
% The purpose of this MATLAB program is to simulate a finite-state semi-Markov
% process.  The process is simulated in order to validate analytical solutions
% for the distribution and moments of the failure time of systems with
% degradation threshold L subject to a SMP environment.  The program uses
% function "rando" in order to select the next state after a state transition.
%
%  Orig Author:  Jeffrey P. Kharoufeh, Ph.D. candidate, IE & OR,
%                Penn State University
%         Date:  January 15, 2001
%   Revised by:  Captain Christopher Solo, M.S., OR,
%                Air Force Institute of Technology
% Last Revised:  January 18, 2004
%*************************************************************************
% VARIABLE DEFINTIONS
%
% WearThreshold  = degradation threshold (system fails after accumulating this
% amount of damage)
% k  = an index variable
%*************************************************************************
% VECTOR/FUNCTION DEFINITIONS
%
% T = Vector of failure time observations
% B = Vector for the initial probability distribution of the semi-Markov process
% P = Probability transition matrix
% V = Vector of degradation rates (i.e., V(i)= degrad. rate when environment is
%     in state i).
% Z = Vector of environment states
%*************************************************************************
%
% The program assumes a state space of the form S={1,2,...K}
%
% Simulation of semi-Markov process (SMP)
%
semi_Markov_input_five;                    % Obtain initialization parameters
Z = [];                                    % Initialize Z.
%
for k = 1:length(Lifetime)
   Z = [];
   Z(1)=rando(B);

   if Z(1) == S(1) |  Z(1) == S(3)
      Lifetime(k) = betarnd(K(Z(1),1),K(Z(1),2)); % Time spent in initial state
   elseif Z(1) == S(2) |  Z(1) == S(4)
      Lifetime(k) = weibrnd(K(Z(1),1),K(Z(1),2));
   elseif Z(1) == S(5)
      Lifetime(k) = gamrnd(K(Z(1),1),K(Z(1),2));
   end
   D = 0.0;                        % At time 0, distance traveled is 0
   D = D + Lifetime(k)*V(Z(1));   % Add traveled distance to cumulative distance
   i=1;
   while D < WearThreshold    % Do while distance traveled is less than WearThr
      Z(i+1) = rando(P(Z(i),:));    % Use the matrix P to determine next state

      if Z(i+1) == S(1) |  Z(i+1) == S(3)
```

```
         new_time = betarnd(K(Z(i+1),1),K(Z(i+1),2));
% Generate beta rv based on current state
      elseif Z(i+1) == S(2) |  Z(i+1) == S(4)
         new_time = weibrnd(K(Z(i+1),1),K(Z(i+1),2));
% Generate beta rv based on current state
      elseif Z(i+1) == S(5)
         new_time = gamrnd(K(Z(i+1),1),K(Z(i+1),2));
% Generate gamma rv based on current state
      end
      D = D + new_time*V(Z(i+1));              % Update cumulative damage
      Lifetime(k) = Lifetime(k) + new_time;    % Update total lifetime
      i=i+1
   end
   Lifetime(k)=Lifetime(k)-(1/V(Z(i)))*(D-WearThreshold);  % Subtract extra time
                                                           % after WearThr is
                                                           % reached
end
get_cdf_five; % computes the empirical distribution of observed failure times
```

```
%*************************************************************************
% PROGRAM semi_Markov_input_five.m
%
% The purpose of this MATLAB program is to provide input data to program
% semi_Markov_five.m for simulation of a finite-state semi-Markov process (K
% states).  The process is simulated in order to validate analytical solutions
% for the distribution and moments of the failure time of systems with
% degradation threshold L subject to a SMP environment.  The program uses
% function "rando" in order to select the next state after a state transition.
%
%  Orig Author:  Jeffrey P. Kharoufeh, Ph.D. candidate, IE & OR,
                 Penn State University
%         Date:  January 15, 2001
%   Revised by:  Captain Christopher Solo, M.S., OR,
                 Air Force Institute of Technology
% Last Revised:  January 17, 2004
%
%*************************************************************************
% VARIABLE DEFINTIONS
*****************************************************************
%
% WearThreshold  = degradation threshold (system fails after accumulating this
%                  amount of damage)
%
%*************************************************************************
% VECTOR/FUNCTION DEFINITIONS
%*********************************************************
%
% T = Vector of failure time observations
% B = Vector for the initial probability distribution of the semi-Markov process
% P = Probability transition matrix
% V = Vector of degradation rates (i.e., V(i)= degrad. rate when environment is
%     in state i).
%*************************************************************************
% Initialize variable and vector values

WearThreshold = 5.0;
N = 5;                   % number of states

Lifetime = zeros(1,100000);     % number of simulations (failure time obs)
B = [1 zeros(1,N-1)];
%
S = [1 2 3 4 5];                   % states of the environment
V = [0.46 0.82 1.10 1.34 1.98];    % matrix of degradation rates
%
P = [0 .5 .2 .2 .1;   % transition probability matrix;
    .5  0 .3 .1 .1;
    .2 .4  0 .2 .2;
    .1 .1 .2  0 .6;
    .2 .1 .1 .6  0];

K = [3 5;     % Parameter matrix--each row gives parameters of distribution
     5 2;
     4 6;
     6 3
    .5 .10];
```

```
%*************************************************************************
% PROGRAM get_cdf_five.m
%
% The purpose of this MATLAB program is to construct an empirical distribution
% function based upon a vector of observations of some continuous random
% variable.  The values of the CDF are generally used for the purpose of
% performing hypothesis test on the equality of two nonparametric distributions.
% Other approaches for testing may be the Cramer von Mises test, which is
% similar to the K-S two-sample test, but slightly more robust.
%
%  Orig Author:  Jeffrey P. Kharoufeh, Ph.D. candidate, IE & OR,
%                Penn State University
%         Date:  June 2000
%   Revised by:  Captain Chris Solo, M.S. candidate, OR,
%                Air Force Institute of Technology
% Last Revised:  January 18, 2004
%*************************************************************************
% VECTOR DEFINITIONS
%*********************************************************************
%
% Lifetime = The finite-dimensional vector of real observations
% t = The vector of points at which to estimate the CDF.
% F = The vector of CDF estimates.  That is, F(i) = F(t(i)), i=1,2,...,nof
%     {1,2,...,C}.
%*************************************************************************
%
F = zeros(1,length(t));  % This ensures that t and F are vectors of equal length
%
% Compute the cdf value at the point t0
%
for i = 1:length(t)
    F(i) = 0.0;
    for j = 1:length(Lifetime)
        if Lifetime(j)<= t(i)
            F(i) = F(i) + 1/length(Lifetime);
        else
            F(i) = F(i);
        end
    end
end
%
% WRITE OUTPUTS TO A TAB DELIMITED TEXT FILE
%
fid = fopen('F:\AFIT\Thesis\Solo Thesis\SMP
simulation\results\chemsimulated.out','w');
for b=1:length(t)
   fprintf(fid, '%4.6f\t%4.6f\n',t(b),F(b));
end
fclose(fid)
```

```
%************************************************************************
% SMPfivestates.m
%
% The purpose of this MATLAB program is to simulate a semi-Markov process (SMP)
% on a state space S and convert it into a continuous-time Markov chain, using a
% phase-type distribution to approximate the holding time distribution of each
% environment state.  Each of the environment states is known to have a Weibull
% holding time distribution (each with different parameters.)  The output
% is the analytical solution for the failure-time distribution of a system
% subject to the SMP environment, based on the results of Kharoufeh (2003) and
% Kharoufeh and Sipe (2004).
%
% Original code for process.m and rando.m obtained from Craig L. Zirbel at
% http://www-math.bgsu.edu/~zirbel/ap/  It appeared that the code was based
% on a discrete time Markov chain, and Captain Steve Cox modified it to work
% for a continuous-time Markov chain.  Captain Chris Solo modified it to be
% a SMP (January 2004).
%
%************************************************************************
%
%       Author:  Captain Christopher J. Solo, M.S. candidate,
%                Operations Research
%                Air Force Institute of Technology
%         Date:  September 13, 2003
% Last Revised:  January 18, 2004
%
%************************************************************************
%                              USER INPUTS
% Tmax = time of SMP run
% numruns = number of simulations (runs)
% mu = initial probability vector of environment states
% S = state space vector
% R = degradation rate vector of environment states
% P = transition probability matrix
% K = matrix of Weibull parameters for environment states
% M = large value used in Big M (hot potato) method
%
% Additionally, rando.m, phasetypeapproxmixed.m, PHvalues.m, invt_lap_Solo.m,
% and failLST.m are needed.
%
%************************************************************************
% For many short runs, set numruns high and Tmax low.  For one long run,
% set numruns equal to 1 and make Tmax high.

format long g;

%*********************************************************************
%                            User Inputs
%
numruns = 1;       % the number of times (runs) to simulate the process

Tmax = 40000;     % This is the length of the run of the environment process

mu = [1,0,0,0,0];    % Semi-Markov process (SMP) starts in state 1

S =  [1,2,3,4,5];    % There are three states defined
```

```
R =  [2.3, 4.1, 5.5, 6.7, 9.9];  % Degradation rate of each state of environment

P = [0 .5 .2 .2 .1;   % transition probability matrix;
    .5  0 .3 .1 .1;
    .2 .4  0 .2 .2;
    .1 .1 .2  0 .6;
    .2 .1 .1 .6  0];

K = [3 5;          % Parameter matrix--each row gives parameters of different
distribution
     5 2;
     4 6;
     6 3
    .5 .10];
%
%*****************************************************************
%
% Simulation of the semi-Markov process (SMP)
%
statetimeTij = zeros(length(S),length(S));
statetimeTi = [];
scale = 1;
%
TimeToFail(1) = 0;       % start times at 0
%Tall(runnum,1) = 0;
x(1) = rando(mu);        % determine starting state (time 0, not time 1)
%xall(runnum,1) = x(1);
rates(1) = R(x(1));
%ratesall(runnum,1) = rates(1);
i = 1;
%
while TimeToFail(i) < Tmax,
    x(i+1) = rando(P(x(i),:));              % Row vector from P-matrix
determines next transitions
    %xall(runnum,i+1) = x(i+1);
    if x(i) == S(1) |  x(i) == S(3)
        time(i) = betarnd(K(x(i),1),K(x(i),2));
% Generate beta rv based on current state
    elseif x(i) == S(2) |  x(i) == S(4)
        time(i) = weibrnd(K(x(i),1),K(x(i),2));
% Generate Weibull rv based on current state
    elseif x(i) == S(5)
        time(i) = gamrnd(K(x(i),1),K(x(i),2));
% Generate gamma rv based on current state
    end
    statetimeTij(x(i),x(i+1)) = statetimeTij(x(i),x(i+1)) + time(i);
    TimeToFail(i+1) = TimeToFail(i) + time(i);  % Add to current time
    %Tall(runnum,i+1) = T(i+1);
    rates(i+1) = R(x(i+1));     % Determine the rate needed for this transition
    %ratesall(runnum,i+1) = rates(i+1);
    i=i+1;
end
%
% Determine the number of transitions from one state to the next
%
numij = zeros(length(S),length(S));
for k = 1:length(x)-2
```

```
    numij(x(k),x(k+1)) = numij(x(k),x(k+1)) + 1;
end
%
% Compute the observed transition rates among the environment states
%
for i = 1:length(numij)
        for j = 1:length(numij)
            if j ~= i
                Qhat(i,j) = numij(i,j)/sum(statetimeTij(i,:));
            else
                Qhat(i,j) = 0;
            end
        end
        Qhat(i,i) = -sum(Qhat(i,:));
        statetimeTi(i,1) = sum(statetimeTij(i,:)); % total time spent in state i
    end
%
%************************************************************************
% This section creates vectors of the holding times in States 1,2, and 3
%  ****IF ADDITIONAL STATES ARE ADDED, THEY MUST BE HARDCODED HERE****
%
for i = 1:length(time)
    for j = 1:length(S)
        if rates(i) == R(j)
           HoldingTime(i,j) = time(i);   % creates matrix where column i
        end                              % represents holding times in state i
    end
end
State1times = HoldingTime(:,1);
State1times = State1times(find(State1times));
% eliminates zeros in holding time column
State2times = HoldingTime(:,2);
State2times = State2times(find(State2times));
% eliminates zeros in holding time column
State3times = HoldingTime(:,3);
State3times = State3times(find(State3times));
% eliminates zeros in holding time column
State4times = HoldingTime(:,4);
State4times = State4times(find(State4times));
% eliminates zeros in holding time column
State5times = HoldingTime(:,5);
State5times = State5times(find(State5times));
% eliminates zeros in holding time column
%
%************************************************************************
% This section computes the phase-type approximations for each of the
environment states
%    ****IF ADDITIONAL STATES ARE ADDED, THEY MUST BE HARDCODED HERE****
%
M=10000;     % this is the large value for the 'hot potato method'
             % i.e. since there is really no 'absorbing phase' for any
             % state, we wish to visit each absorbing phase
             % instantaneously.
             % therefore, its rate is extremely large (approaches infinity)
%

for State = 1:length(S)
```

```
    clear A;
    clear alpha1;
    clear alpha2;
    clear alpha;
    clear beta;
    clear Aparam;
    clear Bparam;
    clear k;
    clear T;
    clear To;
    if State == 1
        distribution = 3;
        A = State1times;
        alpha1=K(State,1);
        alpha2=K(State,2);
        alpha=0; % parameter not needed; dummy value only
        beta=0;  % parameter not needed; dummy value only
        Aparam=0; % parameter not needed; dummy value only
        Bparam=0; % parameter not needed; dummy value only
        [k,T,To] =
phasetypeapproxmixed(A,alpha,beta,alpha1,alpha2,Aparam,Bparam,distribution);
% creates phase-type approximations to each of the holding time distributions
        k1=k;
        T1=T;
        To1=To;
        zerovector1=zeros(1,k1);
        Psimatrix11 = [     T1      To1;      % subgenerator matrix for state 1
                    zerovector1  Qhat(1,1)*M];
    elseif State == 2
        distribution = 2;
        A = State2times;
        alpha1=0; % parameter not needed; dummy value only
        alpha2=0; % parameter not needed; dummy value only
        alpha=K(State,2);
        beta=K(State,1);
        Aparam=0; % parameter not needed; dummy value only
        Bparam=0; % parameter not needed; dummy value only
        [k,T,To] =
phasetypeapproxmixed(A,alpha,beta,alpha1,alpha2,Aparam,Bparam,distribution);
% creates phase-type approximations to each of the holding time distributions
        k2=k;
        T2=T;
        To2=To;
        zerovector2=zeros(1,k2);
        Psimatrix22 = [     T2     To2;      % subgenerator matrix for state 2
                    zerovector2  Qhat(2,2)*M];
    elseif State == 3
        distribution = 3;
        A = State3times;
        alpha=0; % parameter not needed; dummy value only
        beta=0;  % parameter not needed; dummy value only
        alpha1=K(State,1);
        alpha2=K(State,2);
        Aparam=0; % parameter not needed; dummy value only
        Bparam=0; % parameter not needed; dummy value only
        [k,T,To] =
phasetypeapproxmixed(A,alpha,beta,alpha1,alpha2,Aparam,Bparam,distribution);
```

```
% creates phase-type approximations to each of the holding time distributions
        k3=k;
        T3=T;
        To3=To;
        zerovector3=zeros(1,k3);
        Psimatrix33 = [     T3    To3;      % subgenerator matrix for state 3
                    zerovector3  Qhat(3,3)*M];
    elseif State == 4
        distribution = 2;
        A = State4times;
        alpha=K(State,2);
        beta=K(State,1);
        alpha1=0; % parameter not needed; dummy value only
        alpha2=0;  % parameter not needed; dummy value only
        Aparam=0; % parameter not needed; dummy value only
        Bparam=0; % parameter not needed; dummy value only
        [k,T,To] =
phasetypeapproxmixed(A,alpha,beta,alpha1,alpha2,Aparam,Bparam,distribution);
% creates phase-type approximations to each of the holding time distributions
        k4=k;
        T4=T;
        To4=To;
        zerovector4=zeros(1,k4);
        Psimatrix44 = [     T4     To4;      % subgenerator matrix for state 4
                    zerovector4  Qhat(4,4)*M];
    elseif State == 5
        distribution = 5;
        A = State5times;
        Aparam=K(State,1);
        Bparam=K(State,2);
        alpha=0;   % parameter not needed; dummy value only
        beta=0;    % parameter not needed; dummy value only
        alpha1=0;  % parameter not needed; dummy value only
        alpha2=0;  % parameter not needed; dummy value only
        [k,T,To] =
phasetypeapproxmixed(A,alpha,beta,alpha1,alpha2,Aparam,Bparam,distribution);
% creates phase-type approximations to each of the holding time distributions
        k5=k;
        T5=T;
        To5=To;
        zerovector5=zeros(1,k5);
        Psimatrix55 = [     T5      To5;      % subgenerator matrix for state 5
                    zerovector5  Qhat(5,5)*M];
    end
end
%
%***************************************************************************
% OFF-DIAGONAL subgenerator matrices (see p.3-15 of thesis)
% ****IF ADDITIONAL STATES ARE ADDED, THEY MUST BE HARDCODED HERE****
%
Psimatrix12 = zeros(k1+1,k2+1);
Psimatrix12(k1+1,1)=Qhat(1,2)*M;
%
Psimatrix13 = zeros(k1+1,k3+1);
Psimatrix13(k1+1,1)=Qhat(1,3)*M;
%
Psimatrix14 = zeros(k1+1,k4+1);
```

```
Psimatrix14(k1+1,1)=Qhat(1,4)*M;
%
Psimatrix15 = zeros(k1+1,k5+1);
Psimatrix15(k1+1,1)=Qhat(1,5)*M;
%
Psimatrix21 = zeros(k2+1,k1+1);
Psimatrix21(k2+1,1)=Qhat(2,1)*M;
%
Psimatrix23 = zeros(k2+1,k3+1);
Psimatrix23(k2+1,1)=Qhat(2,3)*M;
%
Psimatrix24 = zeros(k2+1,k4+1);
Psimatrix24(k2+1,1)=Qhat(2,4)*M;
%
Psimatrix25 = zeros(k2+1,k5+1);
Psimatrix25(k2+1,1)=Qhat(2,5)*M;
%
Psimatrix31 = zeros(k3+1,k1+1);
Psimatrix31(k3+1,1)=Qhat(3,1)*M;
%
Psimatrix32 = zeros(k3+1,k2+1);
Psimatrix32(k3+1,1)=Qhat(3,2)*M;
%
Psimatrix34 = zeros(k3+1,k4+1);
Psimatrix34(k3+1,1)=Qhat(3,4)*M;
%
Psimatrix35 = zeros(k3+1,k5+1);
Psimatrix35(k3+1,1)=Qhat(3,5)*M;
%
Psimatrix41 = zeros(k4+1,k1+1);
Psimatrix41(k4+1,1)=Qhat(4,1)*M;
%
Psimatrix42 = zeros(k4+1,k2+1);
Psimatrix42(k4+1,1)=Qhat(4,2)*M;
%
Psimatrix43 = zeros(k4+1,k3+1);
Psimatrix43(k4+1,1)=Qhat(4,3)*M;
%
Psimatrix45 = zeros(k4+1,k5+1);
Psimatrix45(k4+1,1)=Qhat(4,5)*M;
%
Psimatrix51 = zeros(k5+1,k1+1);
Psimatrix51(k5+1,1)=Qhat(5,1)*M;
%
Psimatrix52 = zeros(k5+1,k2+1);
Psimatrix52(k5+1,1)=Qhat(5,2)*M;
%
Psimatrix53 = zeros(k5+1,k3+1);
Psimatrix53(k5+1,1)=Qhat(5,3)*M;
%
Psimatrix54 = zeros(k5+1,k4+1);
Psimatrix54(k5+1,1)=Qhat(5,4)*M;
%
Psimatrix = [Psimatrix11 Psimatrix12 Psimatrix13 Psimatrix14 Psimatrix15;
             Psimatrix21 Psimatrix22 Psimatrix23 Psimatrix24 Psimatrix25;
             Psimatrix31 Psimatrix32 Psimatrix33 Psimatrix34 Psimatrix35;
             Psimatrix41 Psimatrix42 Psimatrix43 Psimatrix44 Psimatrix45;
```

```
            Psimatrix51 Psimatrix52 Psimatrix53 Psimatrix54 Psimatrix55];
%
%*************************************************************************
%      This section creates the expanded degradation rate matrix Lambda
%  ****IF ADDITIONAL STATES ARE ADDED, THEY MUST BE HARDCODED HERE****
%
V=diag(R);
L11 = diag(diag(V(1,1)*ones(k1+1)));
L12 = zeros(k1+1,k2+1);
L13 = zeros(k1+1,k3+1);
L14 = zeros(k1+1,k4+1);
L15 = zeros(k1+1,k5+1);
L21 = zeros(k2+1,k1+1);
L22 = diag(diag(V(2,2)*ones(k2+1)));
L23 = zeros(k2+1,k3+1);
L24 = zeros(k2+1,k4+1);
L25 = zeros(k2+1,k5+1);
L31 = zeros(k3+1,k1+1);
L32 = zeros(k3+1,k2+1);
L33 = diag(diag(V(3,3)*ones(k3+1)));
L34 = zeros(k3+1,k4+1);
L35 = zeros(k3+1,k5+1);
L41 = zeros(k4+1,k1+1);
L42 = zeros(k4+1,k2+1);
L43 = zeros(k4+1,k3+1);
L44 = diag(diag(V(4,4)*ones(k4+1)));
L45 = zeros(k4+1,k5+1);
L51 = zeros(k5+1,k1+1);
L52 = zeros(k5+1,k2+1);
L53 = zeros(k5+1,k3+1);
L54 = zeros(k5+1,k4+1);
L55 = diag(diag(V(5,5)*ones(k5+1)));
%
L = [L11 L12 L13 L14 L15;  % expanded degradation rate matrix
     L21 L22 L23 L24 L25;
     L31 L32 L33 L34 L35;
     L41 L42 L43 L44 L45;
     L51 L52 L53 L54 L55];
%
%*************************************************************************
% Display the results of the phase-type approximations
%
fprintf(' Size of generator matrix (Psimatrix):  %g x %g \n', length(Psimatrix),
length(Psimatrix))
fprintf('\n')
fprintf(' Size of expanded degradation rate matrix (Lambda):  %g x %g \n',
length(L), length(L))
fprintf('\n')
fprintf(' ***************************** \n')
fprintf('\n')
%
%*************************************************************************
% This section computes and plots the failure time distribution of the
% system exposed to the SMP environment
%
for w = 1:length(t)
    FailureTimeProb(w) = invt_lap_Solo(t(w),Psimatrix,L);
```

```
    if FailureTimeProb(w) > 1
        FailureTimeProb(w) = 1;   % eliminates CDF values > 1
    elseif FailureTimeProb(w) < 0
        FailureTimeProb(w) = 0;   % eliminates CDF values < 0
    end
end
%************************************************************************
%  Compute first and second moment of failure time distribution
%
First_Moment = invt_lap_first_moment(WearThreshold,Psimatrix,L);
%
Second_Moment = invt_lap_second_moment(WearThreshold,Psimatrix,L);
%
%************************************************************************
%
% WRITE OUTPUTS TO A TAB DELIMITED TEXT FILE
%
fid = fopen('f:\AFIT\Thesis\Solo Thesis\MATLAB files\SMP
simulation\results\chemanalytical.out','w');
for k=1:length(t)
   fprintf(fid, '%4.6f\t%4.6f\n',t(k),FailureTimeProb(k));
end
fclose(fid);
```

Appendix D. Shared Code for Examples

```
%************************************************************************
% Plot_SMP_five.m
%
% This MATLAB program runs the semi-Markov process (SMP) simulation
% (semi_Markov_five.m) and then the 'simulation plus phase-type approximations
% plus analytical solution' program SMPfivestates.m.  The CDF values are then
% compared in a plot (Figure 2).
%
%************************************************************************
%
% Author:  Captain Christopher J. Solo, M.S. candidate, Operations Research
%          Air Force Institute of Technology
%   Date:  January 29, 2004
%
%************************************************************************
%
clear all;
%
t=0.05:0.005:10;  % vector of time points at which to evaluate both CDFs
%
semi_Markov_five;   % calls program to simulate semi-Markov process (SMP) and
obtain
%                     failure times
%
SMPfivestates;   % calls program to simulate SMP (long term), compute phase-type
                 % approximations, & produce analytical results of failure time
                 % distribution
%
% F = cdf values from simulation
% FailureTimeProb = cdf values from phase-type analytical solution
%
% Produces plot of simulated failure-time distribution vs. analytical
% CDF derived via conversion of environment process from SMP to CTMC
%
figure(2);
plot(t,F,'r');
hold on;
plot(t,FailureTimeProb,'b');
grid off;
title('Failure-time CDF values:  Simulated vs. Analytical');
xlabel('t');
ylabel('P(T < t)');
legend('Simulated CDF','Analytical CDF',0);
```

```
%*************************************************************************
% PROGRAM semi_Markov_five.m
%
% The purpose of this MATLAB program is to simulate a finite-state semi-Markov
% process.  The process is simulated in order to validate analytical solutions
% for the distribution and moments of the failure time of systems with
% degradation threshold L subject to a SMP environment.  The program uses
% function "rando" in order to select the next state after a state transition.
%
%  Orig Author:  Jeffrey P. Kharoufeh, Ph.D. candidate, IE & OR,
%                Penn State University
%         Date:  January 15, 2001
%   Revised by:  Captain Christopher Solo, M.S., OR,
%                Air Force Institute of Technology
% Last Revised:  January 18, 2004
%*************************************************************************
% VARIABLE DEFINTIONS
%
% WearThreshold  = degradation threshold (system fails after accumulating this
% amount of damage)
% k  = an index variable
%*************************************************************************
% VECTOR/FUNCTION DEFINITIONS
%
% T = Vector of failure time observations
% B = Vector for the initial probability distribution of the semi-Markov process
% P = Probability transition matrix
% V = Vector of degradation rates (i.e., V(i)= degrad. rate when environment is
%     in state i).
% Z = Vector of environment states
%*************************************************************************
%
% The program assumes a state space of the form S={1,2,...K}
%
% Simulation of semi-Markov process (SMP)
%
semi_Markov_input_five;                    % Obtain initialization parameters
Z = [];                                    % Initialize Z.
%
for k = 1:length(Lifetime)
   Z = [];
   Z(1)=rando(B);

   if Z(1) == S(1) |  Z(1) == S(3)
      Lifetime(k) = betarnd(K(Z(1),1),K(Z(1),2)); % Time spent in initial state
   elseif Z(1) == S(2) |  Z(1) == S(4)
      Lifetime(k) = weibrnd(K(Z(1),1),K(Z(1),2));
   elseif Z(1) == S(5)
      Lifetime(k) = gamrnd(K(Z(1),1),K(Z(1),2));
   end
   D = 0.0;                        % At time 0, distance traveled is 0
   D = D + Lifetime(k)*V(Z(1));   % Add traveled distance to cumulative distance
   i=1;
   while D < WearThreshold    % Do while distance traveled is less than WearThr
      Z(i+1) = rando(P(Z(i),:));    % Use the matrix P to determine next state

      if Z(i+1) == S(1) |  Z(i+1) == S(3)
```

```
         new_time = betarnd(K(Z(i+1),1),K(Z(i+1),2));
% Generate beta rv based on current state
      elseif Z(i+1) == S(2) |  Z(i+1) == S(4)
         new_time = weibrnd(K(Z(i+1),1),K(Z(i+1),2));
% Generate beta rv based on current state
      elseif Z(i+1) == S(5)
         new_time = gamrnd(K(Z(i+1),1),K(Z(i+1),2));
% Generate gamma rv based on current state
      end
      D = D + new_time*V(Z(i+1));              % Update cumulative damage
      Lifetime(k) = Lifetime(k) + new_time;    % Update total lifetime
      i=i+1
   end
   Lifetime(k)=Lifetime(k)-(1/V(Z(i)))*(D-WearThreshold);  % Subtract extra time
                                                           % after WearThr is
                                                           % reached
end
get_cdf_five; % computes the empirical distribution of observed failure times
```

```
%*************************************************************************
% phasetypeapproxmixed.m
%
% The purpose of this MATLAB program is to compare the values of an arbitrary
% probability distribution with those of its 2-phase Coxian or k-phase
% generalized Erlang approximation.
%
%       Author:  Captain Christopher J. Solo, M.S. candidate,
%                Operations Research, Air Force Institute of Technology
%         Date:  September 13, 2003
% Last Revised:  January 18, 2004
%   References:  Perros, H.G. (1994). Queueing Networks with Blocking, 18-21.
%                Altiok, T. (1985). On the phase-type approximations of general
%                  distributions, IIE Transactions, Vol. 17, No. 2, 110-116
%                Neuts, M.F. (1981). Matrix-Geometric Solutions in Stochastic
%                  Models, 41-63.
%                Sauer, C.H. and Chandy, K.M. (1975). Approximate analysis of
%                central server models, IBM Journal of Research and Development,
%                Vol. 19, No. 3, 301-313.
%                Cox, S.R. (2003).  Draft MATLAB code, Air Force Institute of
%                  Technology.
%                Law, A.M. and Kelton, W.D. (2000). Simulation Modeling and
%                  Analysis.
%                Wackerly, D., Mendenhall, W., and Scheaffer, R. (2002).
%                  Mathematical Statistics with Applications, 444.
%
%*************************************************************************
%
function [k,T,To]  =
phasetypeapproxmixed(A,alpha,beta,alpha1,alpha2,Aparam,Bparam,distribution)

CDFdistr=[];   % CDF values of probability distribution
pdfdistr=[];   % density values of probability distribution
CDFapprox=[];  % CDF values of 2-phase Coxian approximation
pdfapprox=[];  % density values of 2-phase Coxian approximation
%
%*************************************************************************
%                  INDICATE TYPE OF INPUT DISTRIBTUION (if known)
% 1 = exponential
% 2 = Weibull
% 3 = beta
% 4 = uniform
% 5 = gamma
%
%*************************************************************************
syms x; % symbolic x used in calculation of first three moments
%*************************************************************************
%                       INDICATE INPUT DISTRIBUTION
%*****************************************
%         EXPONENTIAL DISTRIBUTION
%
if distribution == 1
    lambda = 27.74563544200000; % parameter of exponential distribution
    f = lambda*exp(-lambda*x); % exponential pdf
    obs = 2000; % number of observations from distribution
    t = linspace(0,.5,5*obs);
% time points--'obs' points between 0 and 2, including 0
```

```
    %A = RANDOM('exp',1/lambda,obs,1); % generates a 'obs' x 1 vector A of
observations
%
%****************************************
%           WEIBULL DISTRIBUTION
%
elseif distribution == 2
   f = alpha*(beta^(-alpha))*(x^(alpha - 1))*exp(-(x/beta)^alpha); % Weibull pdf
   obs = 200;
   t = linspace(0,1.4,5*obs); % time points--'obs' points between 0 and 2
    %A = RANDOM('weib',beta,alpha,obs,1); % generates 'obs' x 1 vector of obs
%
%****************************************
%            BETA DISTRIBUTION
%
elseif distribution == 3
    BetaParam = exp(gammaln(alpha1)+gammaln(alpha2)-gammaln(alpha1+alpha2));
    f = ((x^(alpha1 - 1))*(1 - x)^(alpha2 - 1))/BetaParam;
% beta pdf from Law and Kelton
    obs = 200;
    t = linspace(0,1,5*obs);
% time points--'obs' points between 0 and 2, including 0
    %A = RANDOM('beta',alpha1,alpha2,5*obs,1);
% generates a 'obs' x 1 vector A of observations
%
%****************************************
%           UNIFORM DISTRBUTION
%
elseif distribution == 4
    f = 1/(theta2 - theta1);
    obs = 2000;
    %A = RANDOM('unif',theta1,theta2,5*obs,1); % generates a 'obs' x 1 vector A
                                               % of observations
%
%****************************************
%           GAMMA DISTRIBUTION
%
elseif distribution == 5
    obs = 2000;
    t = linspace(0,1.4,5*obs);
end
%*************************************************************************
% %                     Moments of given distribution
%               (use this when input distribution is known)
% %
% if (distribution == 1) | (distribution == 2) % exponential or Weibull
%     lowerlimit = 0;
%     upperlimit = inf;                        % upper limit of integration
%     t = linspace(0,2,500); % time points--500 points between 0 and 2,
including 0
% elseif distribution == 3                     % beta distribution
%     lowerlimit = 0;
%     upperlimit = 1;
%     t = linspace(0,2,500);
% elseif distribution == 4
%     lowerlimit = theta1;
%     upperlimit = theta2;
```

```
%     t = linspace(0,theta2 + 4,500);
% end
% %
% m1 = eval(int((x*f),lowerlimit,upperlimit));     % 1st moment of given
                                                   % distribution
% m2 = eval(int(((x^2)*f),lowerlimit,upperlimit)); % 2nd moment of given
                                                   % distribution
% m3 = eval(int(((x^3)*f),lowerlimit,upperlimit)); % 3rd moment of given
                                                   % distribution
%
%************************************************************************
%                     Moments derived from input data
%
%if A is the vector of input values (holding time observations)
for m = 1:length(A)
    cubed(m,1) = (A(m,1))^3;
end
%
m1 = mean(A);
m2 = var(A) + m1^2;
m3 = (1/length(A))*(sum(cubed)); % from Wackerly
%
%************************************************************************
%    Moments derived via successive derivatives of Laplace transform
%          THIS APPLIES ONLY TO THE EXPONENTIAL DISTRIBUTION
%
% m1=1/lambda;
% m2=2/lambda^2;
% m3=6/lambda^3;
%
%************************************************************************
%                    (Squared) Coefficient of Variation
%
c = (m2-m1^2)/m1^2; % squared coefficient of variation = var/mean^2
z = 3*(m2^2);
v = 2*m1*m3;
%
%************************************************************************
%************************************************************************
%                            FOR c > 1
%   approximation is made using the 2-phase Coxian distribution
%
if c > 1
k=2; % default number of phases
phase = '3-moment, 2-phase Coxian';
%************************************************************************
%                 Moments of 2-phase Coxian approximation
%
Y = (6*m1 - (3*m2/m1))/((6*m2^2/4*m1) - m3); % Altiok (1985), p.112
X = (1/m1) + ((m2*Y)/(2*m1));                % Perros (1994), p.27
mu1 = (X + sqrt(X^2 - 4*Y))/2;   % first moment of 2-phase approximation
mu2 = X - mu1;                   % second moment of 2-phase approximation
%
```

```
%************************************************************************
%                         Underlying CTMC
%
a = (mu2/mu1)*((m1*mu1) - 1);  % prob of transition from phase 1 to phase 2
T = [-mu1 a*mu1;               % matrix of rate transitions among 2 phases
      0    -mu2];
To = [(1-a)*mu1; mu2];   % 2x1 vector of rates of transition out of both states
                         % into absorbing state
initialprob = [1 0];    % initial probability vector
ones=[1;1]; % vector of ones
%
%************************************************************************
%       Original distribution and approximation values and plots
%
PHvalues(c,k,T,To,phase,t,initialprob,ones,distribution,alpha,beta,alpha1,alpha2
,Aparam,Bparam); % calls PHvalues.m file
%
%************************************************************************
%************************************************************************
%                         FOR 0.5 <= c <= 1
% use the two-moment approximation from above (see Perros (1994), p.30)
%
elseif (c >= 0.5) & (c <= 1)
k=2; % default number of phases
mu1 = 2/m1;
mu2 = 1/(m1*c);
phase = '2-moment 2-phase Coxian';
%
%************************************************************************
%                         Underlying CTMC
%
a = 1/(2*c);            % probability of transition from phase 1 to phase 2
T = [-mu1 a*mu1;        % matrix of rate transitions among 2 phases
      0    -mu2];
To = [(1-a)*mu1; mu2];   % 2x1 vector of rates of transition out of both states
into absorbing state
initialprob = [1 0];        % initial probability vector
ones = [1; 1];           % vector of ones
%
%************************************************************************
%       Original distribution and approximation values and plots
%
PHvalues(c,k,T,To,phase,t,initialprob,ones,distribution,alpha,beta,alpha1,alpha2
,Aparam,Bparam); % calls PHvalues.m file
%
%************************************************************************
%************************************************************************
%                          FOR c < 0.5
% approximation is made using a generalized Erlang distribution with k phases
%                  (see Perros (1994), p.29-30)
else
% %
% Linear program to determine minimum number of phases in generalized Erlang
% distribution
%
Aeq=[];
beq=[];
```

```
ub=[];
d = [1];  % choose k such that 1/k <= c AND 1/(k-1) >= c
Y = [-1;
     1];
b = [-1/c (1/c)+1];
lb = 0;
k = ceil(linprog(d,Y,b,Aeq,beq,lb,ub)); % chooses first integer greater than
solution
phase = 'k-phase generalized Erlang';
%
%*****************************************************************
%                        Underlying CTMC
%
% prob of transition from phase i to phase i+1 (see Perros (1994), p.29-30)
a = 1 - (((2*k*c)+k-2-(k^2+4-(4*k*c))^(1/2))/(2*(c+1)*(k-1)));
%
mu = (1+((k-1)*a))/m1; % rate of transition
%
%*****************************************************************
%
% This "for" loop creates the kxk matrix T of transition rates
%
T = zeros(k,k); % preallocates zeroes to the matrix
T(1,2)=a*mu;
for (f=1:k)
    for (g=1:k)

        if f == g
            T (f,g) = -mu;  % diagonal elements
        elseif f > g
            T (f,g) = 0;    % below diagonal elements
        elseif g == (f + 1)
            T (f,g) = mu; % just above diagonal elements
        elseif g > (f + 1)
            T (f,g) = 0;    % above diagonal elements
        end

    end
end
T(1,2)=a*mu;
%*****************************************************************
To = zeros(k,1); % pre-allocates zeroes to the vector
ones = zeros(k,1);
initialprob = zeros(1,k);
To(1,1) = (1-a)*mu; % can enter absorbing state from first phase with prob (1-a)
To(k,1) = mu; % rate of transition out of last phase into absorbing state
ones(1,1)=1;
ones(k,1)=1;
for r=2:k-1
    To(r,1) = 0;  % kx1 vector of rates of transition out of phases 2 through k-
                  % 1 into absorbing state
    ones(r,1) = 1;       % kx1 vector of ones (1,1,1,...1)
end
initialprob(1,1) = 1;    % initial probability vector (1,0,0,...0)
for s=2:k
    initialprob(1,s) = 0;
end
```

```
%*****************************************************************
%     Original distribution and approximation values and plots
%
PHvalues(c,k,T,To,phase,t,initialprob,ones,distribution,alpha,beta,alpha1,alpha2
,Aparam,Bparam); % calls PHvalues.m file
%
%*****************************************************************
end
```

```
%*****************************************************************
% PHvalues.m
%
% The purpose of this MATLAB program is to compute the approximated
% CDF values, the exact CDF values, and their differences, as derived
% from the phasetypeapproxmixed.m program.  This program then plots the
% approximated CDF vs. exact CDF values.
%
%       Author:  Captain Christopher J. Solo, M.S. candidate,
%                Operations Research, Air Force Institute of Technology
%         Date:  September 13, 2003
% Last Revised:  January 18, 2004
%   References:  Perros, H.G. (1994). Queueing Networks with Blocking, 18-21.
%                Neuts, M.F. (1981). Matrix-Geometric Solutions in Stochastic
%                  Models, 41-63.
%
%*****************************************************************
%
function [] =
PHvalues(c,k,T,To,phase,t,initialprob,ones,distribution,alpha,beta,alpha1,alpha2
,Aparam,Bparam)
%                         Output Display
%
fprintf('\n')
%fprintf(' State: %g \n', State)
fprintf('\n')
fprintf(' Number of phases:  %g  \n',k)
fprintf('\n')
fprintf(' Type of approximation: %s \n', phase)
fprintf('\n')
fprintf(' Coefficient of variation  c = %f \n', c)
fprintf('\n')
fprintf(' ***************************** \n')
fprintf('\n')

% fprintf('         t       approx CDF      exact CDF      Difference \n')
%
%*****************************************************************
for u=1:length(t)
     if distribution == 1
         CDFdistr(u)  = expcdf(t(u),1/lambda);       % CDF of the distribution
         pdfdistr(u)  = exppdf(t(u),1/lambda);       % pdf of the distribution
         trueCDF(u,1) = t(u);
         trueCDF(u,2) = CDFdistr(u);
     elseif distribution == 2
         CDFdistr(u)  = weibcdf(t(u),beta,alpha);    % CDF of the distribution
         pdfdistr(u)  = weibpdf(t(u),beta,alpha);    % pdf of the distribution
         trueCDF(u,1) = t(u);
         trueCDF(u,2) = CDFdistr(u);
     elseif distribution == 3
         CDFdistr(u)  = betacdf(t(u),alpha1,alpha2); % CDF of the distribution
         pdfdistr(u)  = betapdf(t(u),alpha1,alpha2); % pdf of the distribution
         trueCDF(u,1) = t(u);
         trueCDF(u,2) = CDFdistr(u);
     elseif distribution == 4
         CDFdistr(u)  = unifcdf(t(u),theta1,theta2); % CDF of the distribution
         pdfdistr(u)  = unifpdf(t(u),theta1,theta2); % pdf of the distribution
```

```
        trueCDF(u,1) = t(u);
        trueCDF(u,2) = CDFdistr(u);
    elseif distribution == 5
        CDFdistr(u)  = gamcdf(t(u),Aparam,Bparam); % CDF of the distribution
        pdfdistr(u)  = gampdf(t(u),Aparam,Bparam); % pdf of the distribution
        trueCDF(u,1) = t(u);
        trueCDF(u,2) = CDFdistr(u);
    end
    CDFapprox(u) = 1-initialprob*(expm(T*t(u)))*ones;
% CDF of approximation (see Neuts, 1981)
    pdfapprox(u) = initialprob*((expm(T*(t(u))))*To);
% pdf of approximation (see Perros, 1994)

   % fprintf('     %f     %f       %f        %f\n', t(u), CDFdistr(u),
CDFapprox(u), abs(CDFdistr(u)-CDFapprox(u)))
end
%
%****************************************************************
%                    Kolmogorov-Smirnov test
% (determines if approximated values and exact values come from same
% distribution)
%
H = kstest2(CDFapprox,CDFdistr);
value = zeros(1,length(t));
for j = 1:length(t)
    value(j) = (CDFapprox(j)-CDFdistr(j))^2;
end
%res = 0.5*sum(value)
%H % H = 1 => reject null that both come from same distribution
%c
%k
%phase
%****************************************************************
%          Plot of k-phase PH approx CDF vs. exact CDF
%
figure(1);
plot(t,CDFapprox,'r:');
hold on;
plot(t,CDFdistr,'b');
grid off;
title('CDF values:  Phase-type vs. Input');
xlabel('t');
ylabel('P(T < t)');
legend('Phase-type','Input',0);
%
% ***************************************************************
```

```
%*************************************************************************
% PROGRAM invt_lap_Solo.m
%
% The purpose of this MATLAB program is to approximate the inverse transform of
% a one-dimensional Laplace transform in order to find the moments of the
% probability distribution, G(t). The program is based on the algorithm of Abate
% and Whitt (1995).
%
%  Orig Author: Jeffrey P. Kharoufeh, Ph.D. candidate, IE & OR,
%               Penn State University
%         Date: January 23, 2001
%   Revised by: Captain Chris Solo, M.S. candidate, OR,
%               Air Force Institute of Technology
%         Date: January 29, 2004
%   References: Abate, J. and W. Whitt (1995).  Numerical Inversion of the
%                 Laplace Transform of Probability Distribution. ORSA Journal on
%                 Computing, 7, 36-43.
%
%*************************************************************************

%Initialize variables, set parameters

function f1 = invt_lap_Solo(t,Psimatrix,L) % inputs time vector, generator
                                           % matrix, and degradation matrix
rho=0.8; qx=[0.8]; tx=[0]; m=11; c=[]; ga=8; A=ga*log(10); mm=2^m;
%
for k=0:m
   d=nchoosek(m,k);
   c=[c d];
end
for t = t;
   tx = t;
   ntr=15;
   u=exp(A/2)/t;
   x=A/(2*t);
   h=pi/t;
   su=zeros(m+2);
   sm=failLST(x,0,Psimatrix,L)/2;
   for k=1:ntr
      y=k*h;
      sm=sm+((-1)^k)*failLST(x,y,Psimatrix,L);
   end
   su(1)=sm;
   for k=1:12
      n=ntr+k;
      y=n*h;
      su(k+1)=su(k)+((-1)^n)*failLST(x,y,Psimatrix,L);
   end
   av1=0; av2=0;
   for k=1:12
      av1=av1+c(k)*su(k);
      av2=av2+c(k)*su(k+1);
   end
   f1 = u*av1/mm; f2=u*av2/mm; qx=[qx f2];
end
```

```
%  Orig Author:  Jeffrey P. Kharoufeh, Ph.D. candidate, IE & OR,
%                Penn State University
%         Date:  June 2000
%*********************************************
function eq=failLST(x,y,Psimatrix,L)

% This program computes the LST of the failure time according to
% Kharoufeh's paper, Sipe's results

s = x+y*i;
I=eye(size(Psimatrix));            % Create identity matrix
A1=inv(L)*((Psimatrix-(s*I))*5.0); % The last argument is the degradation
                                   % threshold level, x.
A2=expm(A1);
J = zeros(1,(length(Psimatrix)-1)); % total number of phases minus one = number
                                    % of zeros to use
alphavec = [1 J];   % initial probability vector:  Ensure the # columns = #
                    % states (phases).
m=ones(size(Psimatrix,1),1);
z = (1/s)*(alphavec*A2*m);  % get the cdf
eq = real(z);               % get the cdf
```

```
%*************************************************************************
% PROGRAM invt_lap_first_moment.m
%
% The purpose of this MATLAB program is to approximate the inverse transform of
% a one-dimensional Laplace transform in order to find the moments of the
% probability distribution, G(t). The program is based on the algorithm of Abate
% and Whitt (1995).
%
%  Orig Author: Jeffrey P. Kharoufeh, Ph.D. candidate, IE & OR,
%               Penn State University
%         Date: January 23, 2001
%   Revised by: Captain Chris Solo, M.S. candidate, OR,
%               Air Force Institute of Technology
%         Date: January 29, 2004
%   References: Abate, J. and W. Whitt (1995).  Numerical Inversion of the
%                 Laplace Transform of Probability Distribution. ORSA Journal on
%                 Computing, 7, 36-43.
%
%*************************************************************************
%
%Initialize variables, set parameters

function f1 = invt_lap_first_moment(t,Psimatrix,L) % inputs time vector,
                                                   % generator matrix,
                                                   % and degradation matrix
rho=0.8; qx=[0.8]; tx=[0]; m=11; c=[]; ga=8; A=ga*log(10); mm=2^m;
%
for k=0:m
   d=nchoosek(m,k);
   c=[c d];
end
for t = t;
   tx = t;
   ntr=15;
   u=exp(A/2)/t;
   x=A/(2*t);
   h=pi/t;
   su=zeros(m+2);
   sm=first_momentLST(x,0,Psimatrix,L)/2;
   for k=1:ntr
      y=k*h;
      sm=sm+((-1)^k)*first_momentLST(x,y,Psimatrix,L);
   end
   su(1)=sm;
   for k=1:12
      n=ntr+k;
      y=n*h;
      su(k+1)=su(k)+((-1)^n)*first_momentLST(x,y,Psimatrix,L);
   end
   av1=0; av2=0;
   for k=1:12
      av1=av1+c(k)*su(k);
      av2=av2+c(k)*su(k+1);
   end
   f1 = u*av1/mm; f2=u*av2/mm; qx=[qx f2];
end
```

```
%*************************************************************************
% PROGRAM invt_lap_second_moment.m
%
% The purpose of this MATLAB program is to approximate the inverse transform of
% a one-dimensional Laplace transform in order to find the moments of the
% probability distribution, G(t). The program is based on the algorithm of Abate
% and Whitt (1995).
%
%  Orig Author: Jeffrey P. Kharoufeh, Ph.D. candidate, IE & OR,
%               Penn State University
%         Date: January 23, 2001
%   Revised by: Captain Chris Solo, M.S. candidate, OR,
%               Air Force Institute of Technology
%         Date: January 29, 2004
%   References: Abate, J. and W. Whitt (1995).  Numerical Inversion of the
%                 Laplace Transform of Probability Distribution. ORSA Journal on
%                 Computing, 7, 36-43.
%
%*************************************************************************
%
%Initialize variables, set parameters

function f1 = invt_lap_second_moment(t,Psimatrix,L) % inputs time vector,
                                                    % generator matrix,
                                                    % and degradation matrix
rho=0.8; qx=[0.8]; tx=[0]; m=11; c=[]; ga=8; A=ga*log(10); mm=2^m;
%
for k=0:m
   d=nchoosek(m,k);
   c=[c d];
end
for t = t;
   tx = t;
   ntr=15;
   u=exp(A/2)/t;
   x=A/(2*t);
   h=pi/t;
   su=zeros(m+2);
   sm=second_momentLST(x,0,Psimatrix,L)/2;
   for k=1:ntr
      y=k*h;
      sm=sm+((-1)^k)*second_momentLST(x,y,Psimatrix,L);
   end
   su(1)=sm;
   for k=1:12
      n=ntr+k;
      y=n*h;
      su(k+1)=su(k)+((-1)^n)*second_momentLST(x,y,Psimatrix,L);
   end
   av1=0; av2=0;
   for k=1:12
      av1=av1+c(k)*su(k);
      av2=av2+c(k)*su(k+1);
   end
   f1 = u*av1/mm; f2=u*av2/mm; qx=[qx f2];
end
```

```
%*************************************************************************
% PROGRAM first_momentLST.m
%
% This program computes the LST of the first moment according to
% Kharoufeh (2003) and Kharoufeh and Sipe (2004).
%
%  Orig Author:  Jeffrey P. Kharoufeh, Ph.D. candidate, IE & OR,
%                Penn State University
%   Revised by:  Captain Christopher Solo, M.S. candidate, OR,
%                Air Force Institute of Technology
% Last Revised:  March 11, 2004
%*************************************************************************
%
function eq=first_momentLST(x,y,Psimatrix,L)

clear r;
n = 1;
r = x+y*i;
B1=(r*L)-Psimatrix;
J = zeros(1,(length(Psimatrix)-1)); % total number of phases minus one = number
                                    % of zeros to use
alphavec = [1 J];   % initial probability vector:  Ensure the # columns = #
                    % states (phases).
m=ones(size(Psimatrix,1),1);
z = (1/r)*(factorial(n))*alphavec*((B1)^(-n))*m;  % get the first moment
eq = real(z);          % get the cdf
```

```
%*************************************************************************
% PROGRAM second_momentLST.m
%
% This program computes the LST of the second moment according to
% Kharoufeh (2003) and Kharoufeh and Sipe (2004).
%
%  Orig Author:  Jeffrey P. Kharoufeh, Ph.D. candidate, IE & OR,
%                Penn State University
%   Revised by:  Captain Christopher Solo, M.S. candidate, OR,
%                Air Force Institute of Technology
% Last Revised:  March 11, 2004
%*************************************************************************
%
function eq=second_momentLST(x,y,Psimatrix,L)

% This program computes the LST of the failure time according to
% Kharoufeh's paper, Sipe's results
clear r;
n = 2;
r = x+y*i;
B1=(r*L)-Psimatrix;
J = zeros(1,(length(Psimatrix)-1)); % total number of phases minus one = number
                                    % of zeros to use
alphavec = [1 J];   % initial probability vector:  Ensure the # columns = #
                    % states (phases).
m=ones(size(Psimatrix,1),1);
z = (1/r)*(factorial(n))*alphavec*((B1)^(-n))*m;  % get the first moment
eq = real(z);          % get the cdf
```

```
%*******************************************************************************
% PROGRAM rando.m
%
% This program generates a random variable in 1,2,...,n given a distribution
% vector.
%
% Orig Author:  Jeffrey P. Kharoufeh, Ph.D. candidate, IE & OR,
%               Penn State University
%        Date:  June 2000
%*******************************************************************************
%
function [index] = rando(p)             % Example x(1) = rando(mu) then p = mu
u = rand;
i = 1;
s = p(1);                                   % s = first probability in p

while ((u > s) & (i < length(p))),          % Random draw compared to
probability
  i=i+1;
  s=s+p(i);                             % Must be a stochastic vector (sum = 1)
end

index=i;                                % Returns state to transition to next
```

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From – To)*
12-03-2004	**Master's Thesis**	Jul 2003 – Mar 2004

4. TITLE AND SUBTITLE

PHASE-TYPE APPROXIMATIONS FOR WEAR PROCESSES IN A SEMI-MARKOV ENVIRONMENT

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Solo, Christopher J., Captain, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Street, Building 642
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GOR/ENS/04-11

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
USSTRATCOM/PR12
Attn: Dr. Mark A. Gallagher
901 SAC BLVD, SUITE 2F16 phone: DSN 271-1938
Offutt AFB, NE 68113-6500 e-mail: gallaghm@stratcom.mil

10. SPONSOR/MONITOR'S ACRONYM(S)
USSTRATCOM/PR12

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The reliability of a single-unit system experiencing degradation (wear) due to the influence of a general, observable environment process is considered. In particular, the failure time distribution is evaluated using only observations of the unit's current operating environment which is characterized as a finite semi-Markov process (SMP). In order to impose the Markov property, generally distributed environment state sojourn times are approximated as phase-type (PH) random variables using observations of state holding times and transition rates. The use of PH distributions facilitates the use of existing analytical results for reliability evaluation of units subject to an environment process that evolves as a continuous-time Markov chain. The procedure is illustrated through three numerical examples, and results are compared with those obtained via Monte Carlo simulation. The maximum absolute deviation in probability for failure time distributions was on the order of 0.004. The results of this thesis provide a novel approach to the reliability analysis of units operating in randomly evolving environments for which degradation or failure time observations are difficult or impossible to obtain.

15. SUBJECT TERMS
phase-type, semi-Markov, wear process, degradation, failure time

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Jeffrey P. Kharoufeh, Ph.D. (ENS)
U	U	U	UU	138	19b. TELEPHONE NUMBER *(Include area code)* (937) 255-3636 ext 4603; e-mail: Jeffrey.Kharoufeh@afit.edu